K 1

```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
```

```
_$

Va
---
000
000
000
7F
7F
7F
7F
7F
7F
7F
7F
7F
```

K 13

```
**FILE**ID**UETTAPE00
```

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  TTTTTTTTTT    AAAAAA    PPPPPPPP    EEEEEEEEEE    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  TTTTTTTTTT    AAAAAA    PPPPPPPP    EEEEEEEEEE    000000      000000
UU      UU  EE              TT          TT      AA      AA  PP      PP  EE          00      00  00      00
UU      UU  EE              TT          TT      AA      AA  PP      PP  EE          00      00  00      00
UU      UU  EE              TT          TT      AA      AA  PP      PP  EE          00    0000  00    0000
UU      UU  EE              TT          TT      AA      AA  PP      PP  EE          00    0000  00    0000
UU      UU  EEEEEEEE        TT          T       AA      AA  PPPPPPPP    EEEEEEEE    00  00  00  00  00  00
UU      UU  EEEEEEEE        TT          TT      AA      AA  PPPPPPPP    EEEEEEEE    00  00  00  00  00  00
UU      UU  EE              TT          TT      AAAAAAAAAA  PP          EE          0000    00  0000    00
UU      UU  EE              TT          TT      AAAAAAAAAA  PP          EE          0000    00  0000    00
UU      UU  EE              TT          TT      AA      AA  PP          EE          00      00  00      00      ....
UU      UU  EE              TT          TT      AA      AA  PP          EE          00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT          TT      AA      AA  PP          EEEEEEEEEE    000000      000000        ....
UUUUUUUUUU  EEEEEEEEEE      TT          TT      AA      AA  PP          EEEEEEEEEE    000000      000000        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
    (2)      111        Declarations
    (3)      226        Read-Only Data
    (4)      392        Read/Write Data
    (5)      598        RMS-32 Data Structures
    (6)      667        Main Program
    (12)    1064        Test the Magtape
    (17)    1528        Mount Routine
    (18)    1710        Dismount Routine
    (19)    1831        Initialize Routine
    (20)    2067        Timer Expiration Routine
    (21)    2135        System Service Exception Handler
    (22)    2265        RMS Error Handler
    (23)    2369        CTRL/C Handler
    (24)    2418        Error Exit
    (25)    2480        Exit Handler
```

```
0000             .TITLE UETTAPE00 VAX/VMS UETP DEVICE TEST FOR TAPE
0000             .IDENT  'V04-000'
0000             .ENABLE SUPPRESSION
0000       ;
0000       ;****************************************************************************
0000       ;*                                                                          *
0000       ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000       ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000       ;*  ALL RIGHTS RESERVED.                                                    *
0000       ;*                                                                          *
0000       ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000       ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000       ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000       ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000       ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000       ;*  TRANSFERRED.                                                            *
0000       ;*                                                                          *
0000       ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000       ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000       ;*  CORPORATION.                                                            *
0000       ;*                                                                          *
0000       ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000       ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000       ;*                                                                          *
0000       ;*                                                                          *
0000       ;****************************************************************************
0000       ;
0000
0000       ;++
0000       ; FACILITY:
0000       ;        This module will be distributed with VAX/VMS under the [SYSTEST]
0000       ;        account.
0000       ;
0000       ; ABSTRACT:
0000       ;        This program tests all supported mag tapes.  It uses QIO's in the
0000       ;        ONE PASS mode and RMS block mode with variable record size in the
0000       ;        NORMAL and LOOP modes. Tapes are rewound, dismounted and initialized
0000       ;        on exit.
0000       ;
0000       ; ENVIRONMENT:
0000       ;        This program will run in user access mode, with AST's enabled except
0000       ;        during error processing. This program requires the following privileges
0000       ;        and quotas.
0000       ;
0000       ;                GRPNAM, LOG_IO
0000       ;                AST queue = 2 + number of tape units under test
0000       ;
0000       ;
0000       ; AUTHOR: Robert N. Perron              CREATION DATE: Feb., 1981
0000       ;
0000       ; MODIFIED BY:
0000       ;
0000       ;        V03-007 RNH0006         Richard N. Holstein,    01-Jul-1984
0000       ;                Make one-shot timer more forgiving.  Explicitly deassign tape
0000       ;                if we get an error in one-shot mode.
0000       ;
0000       ;        V03-006 RNH0005         Richard N. Holstein,    15-Feb-1984
```

```
0000      ;               Take advantage of new UETP message codes.  Fix SSERROR
0000      ;               interaction with RMS_ERROR.
0000      ;
0000      ;       V03-005 RNH0004         Richard N. Holstein,    19-Dec-1983
0000      ;               Give correct sentinels to Test Controller.
0000      ;
0000      ;       V03-004 RNH0003         Richard N. Holstein,    11-Mar-1983
0000      ;               Don't signal ending message in EXIT_HANDLER.
0000      ;
0000      ;       V03-003 RNH0002         Richard N. Holstein,    25-Feb-1983
0000      ;               Allow for longer device names.
0000      ;
0000      ;       V03-002 RNP0009         Robert N. Perron,       08-Nov-1982
0000      ;               Add code so that tape init routine reuses the same
0000      ;               termination mbx each pass in loop mode. Also restore
0000      ;               SSFM and AST enable mode before exiting dismount routine.
0000      ;
0000      ;       V03-001 RNH0001         Richard N. Holstein,    15-Oct-1982
0000      ;               Miscellaneous fixes listed in the V3B UETP Workplan.
0000      ;
0000      ;       V02-009 RNP0008         Robert N. Perron,       02-Mar-1982
0000      ;               Enable loop mode.
0000      ;
0000      ;       V02-008 RNP0007         Robert N. Perron,       23-Jan-1982
0000      ;               Changed to conform to new mount system service interface.
0000      ;
0000      ;       V02-007 RNP0006         Robert N. Perron,       17-Nov-1981
0000      ;               Activated code to utilize mount system service.
0000      ;               Changed the .ENTRY and .TITLE to UETTAPE00.
0000      ;
0000      ;       V02-006 RNP0005         Robert N. Perron,       28-Sep-1981
0000      ;               Changed TEST_NAME to agree with UETSUPDEV.DAT.
0000      ;               Changed watch dog timers from using event flag #0 (default).
0000      ;
0000      ;       V02-005 RNP0004         Robert N. Perron,       22-Sep-1981
0000      ;               Changed INIT process to be detached instead of sub.
0000      ;
0000      ;       V02-004 LDJ0001         Larry D. Jones,         21-Sep-1981
0000      ;               Changed the .ENTRY and .TITLE to be UETTAPE01
0000      ;
0000      ;       V02-003 RNP0003         Robert N. Perron,       14-Sep-1981
0000      ;               Increased dismount watch dog timer interval.
0000      ;
0000      ;       V02-002 RNP0002         Robert N. Perron,       11-Sep-1981
0000      ;               Fixed race condition between dismount and init routines.
0000      ;
0000      ;       V02-001 RNP0001         Robert N. Perron,       02-Sep-1981
0000      ;               Modified so that UETINIDEV.DAT is updated only when in
0000      ;               oneshot mode.
0000      ;
0000      ;**
0000      ;--
```

```
          0000          .SBTTL  Declarations
          0000  ;
          0000  ; INCLUDE FILES:
          0000  ;
          0000  ;       SYS$LIBRARY:LIB.MLB      for general definitions
          0000  ;       SHRLIB$:UETP.MLB         for UETP definitions
          0000
          0000  ;
          0000  ; MACROS:
          0000  ;
          0000          $ACCDEF                         ; Accounting definitions
          0000          $CHFDEF                         ; Condition handler frame definitions
          0000          $DEVDEF                         ; Device definitions
          0000          $DIBDEF                         ; Device Information Block
          0000          $DMTDEF                         ; Dismount system service definitions
          0000          $DVIDEF                         ; $GETDVI ITMLST item codes
          0000          $FIBDEF                         ; Define file info block symbols
          0000          $IODEF                          ; Define I/O function codes
          0000          $JPIDEF                         ; Getjpi definitions
          0000          $MNTDEF                         ; Mount system sevice definitions
          0000          $MTDEF                          ; Magtape definitions
          0000          $SHRDEF                         ; Shared messages
          0000          $SSDEF                          ; System Service status codes
          0000          $STSDEF                         ; Status return
          0000          $UETUNTDEF                      ; UETP unit block offset definitions
          0000          $UETPDEF                        ; UETP
          0000  ;
          0000  ; EQUATED SYMBOLS:
          0000  ;
          0000  ;   Facility number definitions:
00000001  0000          RMS$_FACILITY = 1
          0000
          0000  ;   SHR message definitions:
00740000  0000          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0  0000          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038  0000          UETP$_BEGIND = UETP!SHR$_BEGIND
00741080  0000          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098  0000          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130  0000          UETP$_TEXT   = UETP!SHR$_TEXT
          0000
          0000  ;   Internal flag bits...:
00000001  0000          TEST_OVERV     = 1              ; Set when pass timer expires
00000002  0000          SAFE_TO_UPDV   = 2              ; Set if it's safe to update UETINIDEV
00000003  0000          BEGIN_MSGV     = 3              ; Set if "BEGIN" msg has been printed
00000004  0000          ONESHOT_MODEV = 4               ; Set when "MODE" is "oneshot"
00000005  0000          LOOP_MODEV     = 5              ; Set when "MODE" is "loop"
00000006  0000          DATA_ERRORV    = 6              ; Set when compare of read & write data
          0000                                          ; ...fails in "one shot" mode
00000007  0000          TEST_STARTV    = 7              ; Set when testing is started in normal
          0000                                          ; or loop modes
00000008  0000          MBX_CREATEDV   = 8              ; Set when termination mbx is first created
          0000
          0000  ;   ...and corresponding masks:
00000002  0000          TEST_OVERM      = 1@TEST_OVERV
00000004  0000          SAFE_TO_UPDM    = 1@SAFE_TO_UPDV
00000008  0000          BEGIN_MSGM      = 1@BEGIN_MSGV
00000010  0000          ONESHOT_MODM    = 1@ONESHOT_MODEV
```

```
00000020  0000             LOOP_MODM      = 1@LOOP_MODEV
00000040  0000             DATA_ERRM      = 1@DATA_ERRORV
00000080  0000             TEST_STARTM    = 1@TEST_STARTV
00000100  0000             MBX_CREATEDM   = 1@MBX_CREATEDV
          0000
          0000     ;   Unit block device dependent flag bits:
00000003  0000             UETUNT$V_MOUNTED  = 3           ; Set when tape is mounted
00000004  0000             UETUNT$V_MODIFIED = 4           ; Set if we try to do a CREATE
          0000
          0000     ;   ...and corresponding masks:
00000008  0000             UETUNT$M_MOUNTED  = 1@UETUNT$V_MOUNTED
00000010  0000             UETUNT$M_MODIFIED = 1@UETUNT$V_MODIFIED
          0000
          0000     ;   Miscellany:
00000020  0000             LC_BITM        = ^X20           ; Mask to convert lower case to upper
00000028  0000             REC_SIZE       = 40             ; UETINIDEV.DAT record size
00000084  0000             TEXT_BUFFER    = 132            ; Internal text buffer size
00000001  0000             REQIDT1        = 1              ; AST parameter for pass completion
00000002  0000             REQIDT2        = 2              ; AST parameter for device hung
00000003  0000             SS_SYNCH_EFN   = 3              ; Synch miscellaneous system services
0000000A  0000             MAX_DEV_DESIG = 10              ; Longest possible controller name
00000005  0000             MAX_UNIT_DESIG= 5               ; Longest possible unit number
0000000F  0000             MAX_PROC_NAME = 15              ; Longest possible process name
00000100  0000             MBX_SIZE       = 256            ; Termination mailbox size
00000005  0000             DENS_LEN       = 5              ; Length of density string
0000001B  0000             ESC            = ^X1B           ; Escape character
          0000
          0000     ;
          0000     ;   Device dependent definitions:
          0000     ;
          0000
          0000     ; Orginal tape density
000001A4  0000             UETUNT$K_DENSITY  = UETUNT$K_DEVDEP
          0000     ; Device name descriptor
000001A9  0000             UETUNT$Q_DEVDSC   = UETUNT$K_DEVDEP+DENS_LEN
          0000     ; Device name buffer
000001B1  0000             UETUNT$K_DEV_NAM  = UETUNT$K_DEVDEP+8+DENS_LEN
          0000     ; Index for buffer size list
000001C0  0000             UETUNT$B_BUFPTR   = UETUNT$K_DEVDEP+8+DENS_LEN+MAX_DEV_DESIG+MAX_UNIT_DESIG
          0000     ; Index for density list
000001C1  0000             UETUNT$B_DENSPTR  = UETUNT$K_DEVDEP+9+DENS_LEN+MAX_DEV_DESIG+MAX_UNIT_DESIG
          0000     ; Unit read buffer
000001C2  0000             UETUNT$K_RBUF     = UETUNT$K_DEVDEP+10+DENS_LEN+MAX_DEV_DESIG+MAX_UNIT_DESIG
          0000
          0000     ;   The following definitions are set depending on the device under test.
          0000     ;   (all in bytes)
          0000
0000001E  0000             DEVDEP_SIZE    = 10+DENS_LEN+MAX_DEV_DESIG+MAX_UNIT_DESIG  ; Size of
          0000                                             ; device dependent part of unit block
00008000  0000             WRITE_SIZE     = 32768          ; Size of device write buffer
00008000  0000             READ_SIZE      = 32768          ; Size of device read buffer
          0000
          0000             PAGES = <<UETUNT$C_INDSIZ+-     ; Add together all of the pieces...
          0000                     DEVDEP_SIZE+-           ; ...which make up a UETP unit block...
          0000                     READ_SIZE+-             ; ...to give to the $EXPREG service
00000041  0000                     511>/512>
          0000
```

```
                                   0000    226          .SBTTL  Read-Only Data
                               00000000    227          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                   0000    228
                                   0000    229 ACNT_NAME:                              ; Process name on exit
53 45 54 53 59 53 00000008'010E0000' 0000  230          .ASCID  /SYSTEST/
                              54   000E
                                   000F    231
                                   000F    232 TEST_NAME:                              ; This test name
50 41 54 54 45 55 00000017'010E0000' 000F  233          .ASCID  /UETTAPE00/
                        30 30 45   001D
                                   0020    234
                                   0020    235 SUPDEV_GBLSEC:                          ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000' 0020  236          .ASCID  /UETSUPDEV/
                        56 45 44   002E
                                   0031    237
                                   0031    238 CONTROLLER:                             ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000' 0031  239          .ASCID  /CTRLNAME/
                           45 4D   003F
                                   0041    240
                                   0041    241 MODE:                                   ; Run mode logical name
      45 44 4F 4D 00000049'010E0000' 0041  242          .ASCID  /MODE/
                                   004D    243
                                   004D    244 LABEL:                                  ; Required tape label
20 20 50 54 45 55 00000055'010E0000' 004D  245          .ASCID  /UETP        /         ; 12 characters, same as DIB field
               20 20 20 20 20 20   005B
                                   0061    246
                                   0061    247 NO_RMS_AST_TABLE:                       ; List of errors for which...
                        00000000'  0061    248          .LONG   RMS$_BLN               ; ...RMS cannot deliver an AST...
                        00000000'  0065    249          .LONG   RMS$_BUSY              ; ...even if one has an ERR= arg
                        00000000'  0069    250          .LCNG   RMS$_CDA               ; Note that we can search table...
                        00000000'  006D    251          .LONG   RMS$_FAB               ; ...via MATCHC since <31:16>...
                        00000000'  0071    252          .LONG   RMS$_RAB               ; ...pattern can't be in <15:0>
                        00000014   0075    253 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                   0075    254
                                   0075    255 SYS$INPUT:                              ; Name of device from which...
4E 49 24 53 59 53 0000007D'010E0000' 0075  256          .ASCID  /SYS$INPUT/            ; ...the test can be aborted
                        54 55 50   0083
                                   0086    257
                                   0086    258 INPUT_ITMLST:                           ; $GETDVI arg list for SYS$INPUT
                       0020 0040   0086    259          .WORD   64,DVI$_DEVNAM         ; We need the equivalence name
              0000000C'00000014'   008A    260          .LONG   BUFFER,BUFFER_PTR
                        00000000   0092    261          .LONG   0                      ; Terminate the list
                                   0096    262
                                   0096    263 CS1:                                    ; Device class and type control string
21 20 42 58 32 21 0000009E'010E0000' 0096  264          .ASCID  /!2XB !2XB /
                     20 42 58 32   00A4
                                   00A8    265
                                   00A8    266 CS3:                                    ; Device class-only control string
2A 20 42 58 32 21 000000B0'010E0000' 00A8  267          .ASCID  /!2XB **/
                              2A   00B6
                                   00B7    268
                                   00B7    269 CNTRLCMSG:
65 74 72 6F 62 41 000000BF'010E0000' 00B7  270          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64  00C5
            43 2F 4C 52 54 43 20   00D1
                                   00D8    271
                                   00D8    272 NO_CTRLNAME.
```

```
6E 6F 63 20 6F 4E 000000E0'010E0000'  00D8    273             .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74   00E6
               2E 64 65 69 66 69      00F2
                                      00F8    274
                                      00F8    275 DEAD_CTRLNAME:
20 74 27 6E 61 43 00000100'010E0000'  00F8    276             .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74   0106
72 61 6D 20 2C 53 41 21 20 72 65 6C   0112
61 73 75 6E 75 20 73 61 20 64 65 6B   011E
4E 49 54 45 55 20 6E 69 20 65 6C 62   012A
         2E 54 41 44 2E 56 45 44 49   0136
                                      013F    277
                                      013F    278 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000147'010E0000'  013F    279             .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74   014D
2E 67 6E 69 74 73 65 74 20 72 6F 66   0159
                                      0165    280
                                      0165    281 NOUNIT_TESTABLE:
73 65 74 20 6F 4E 0000016D'010E0000'  0165    282             .ASCID  /No testable units./
2E 73 74 69 6E 75 20 65 6C 62 61 74   0173
                                      017F    283
                                      017F    284 ILLEGAL_REC:
61 67 65 6C 6C 49 00000187'010E0000'  017F    285             .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C   018D
20 65 6C 69 66 20 6E 69 20 74 61 6D   0199
41 44 2E 56 45 44 49 4E 49 54 45 55   01A5
                              21 54   01B1
                                      01B3    286
                                      01B3    287 PASS_MSG:
66 6F 20 64 6E 45 000001BB'010E0000'  01B3    288             .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20   01C1
61 72 65 74 69 20 4C 55 21 20 68 74   01CD
44 25 21 20 74 61 20 73 6E 6F 69 74   01D9
                                 2E   01E5
                                      01E6    289
                                      01E6    290 TIME_OUT_MSG:                         ; Used by one shot mode
6F 20 65 6D 69 54 000001EE'010E0000'  01E6    291             .ASCID  /Time out - drive off line or not testable./
6F 20 65 76 69 72 64 20 2D 20 74 75   01F4
6E 20 72 6F 20 65 6E 69 6C 20 66 66   0200
2E 65 6C 62 61 74 73 65 74 20 74 6F   020C
                                      0218    292
                                      0218    293 DATA_ERR_MSG:
63 20 61 74 61 44 00000220'010E0000'  0218    294             .ASCID  /Data compare error while testing !AS./
72 6F 72 72 65 20 65 72 61 70 6D 6F   0226
69 74 73 65 74 20 65 6C 69 68 77 20   0232
               2E 53 41 21 20 67 6E   023E
                                      0245    295
                                      0245    296 MNT_ERR_MSG:
20 72 6F 72 72 45 0000024D'010E0000'  0245    297             .ASCID  /Error while mounting !AS./
69 74 6E 75 6F 6D 20 65 6C 69 68 77   0253
               2E 53 41 21 20 67 6E   025F
                                      0266    298
                                      0266    299 LABEL_ERR_MSG:
73 27 53 41 21 20 0000026E'010E0000'  0266    300             .ASCID  / !AS's label !AC is incorrect - this test requires !AS./
69 20 43 41 21 20 6C 65 62 61 6C 20   0274
20 74 63 65 72 72 6F 63 6E 69 20 73   0280
20 74 73 65 74 20 73 69 68 74 20 2D   028C
```

```
53 41 21 20 73 65 72 69 75 71 65 72  0298
                                 2E  02A4
                                     02A5     301
                                     02A5     302 HWL_ERR_MSG:
69 20 53 41 21 20 000002AD'010E0000' 02A5     303         .ASCID  / !AS is write-locked./
6B 63 6F 6C 2D 65 74 69 72 77 20 73  02B3
                           2E 64 65  02BF
                                     02C2     304
                                     02C2     305 DENSITY_ERR:
6F 63 65 72 6E 55 000002CA'010E0000' 02C2     306         .ASCID  /Unrecognizable density./
6E 65 64 20 65 6C 62 61 7A 69 6E 67  02D0
                     2E 79 74 69 73  02DC
                                     02E1     307
                                     02E1     308 INIT_ERR_MSG:
20 72 6F 72 72 45 000002E9'010E0000' 02E1     309         .ASCID  /Error while initializing !AS./
61 69 74 69 6E 69 20 65 6C 69 68 77  02EF
   2E 53 41 21 20 67 6E 69 7A 69 6C  02FB
                                     0306     310
                                     0306     311 DISMNT_ERR_MSG:
20 72 6F 72 72 45 0000030E'010E0000' 0306     312         .ASCID  /Error while dismounting !AS./
75 6F 6D 73 69 64 20 65 6C 69 68 77  0314
      2E 53 41 21 20 67 6E 69 74 6E  0320
                                     032A     313
                                     032A     314 INIDEV_UPDERR:                          ; Error during exit handler
20 72 6F 72 72 45 00000332'010E0000' 032A     315         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75  0338
   2E 54 41 44 2E 56 45 44 49 4E 49  0344
                                     034F     316
                                     034F     317 THIRTYSEC:
                           11E1A300  034F     318         .LONG   10*1000*1000*30         ; 30 seconds  time
                                     0353     319
                                     0353     320 THIRTYSEC_DELTA:
                  FFFFFFFF EE1E5D00  0353     321         .LONG   -10*1000*1000*30,-1     ; 30 seconds delta time
                                     035B     322
                                     035B     323 ONEMIN_DELTA:                           ; 1 minute delta time
                  FFFFFFFF DC3CBA00  035B     324         .LONG   -10*1000*1000*60,-1
                                     0363     325
                                     0363     326 THREEMIN:                               ; 3 minutes time
                           6B49D200  0363     327         .LONG   10*1000*1000*180
                                     0367     328
                                     0367     329 THREEMIN_DELTA:                         ; 3 minutes delta time
                  FFFFFFFF 94B62E00  0367     330         .LONG   -10*1000*1000*180,-1
                                     036F     331
                                     036F     332 CONT_DESC:                              ; Descriptor used to convert controller...
                          0000 0028  036F     333         .WORD   REC_SIZE,0              ; ...from lowercase to uppercase
                          00000014'  0373     334         .ADDRESS BUFFER
                                     0377     335
                                     0377     336 RMS_ERR_MSG:                            ; Announces an RMS error
72 65 20 53 4D 52 0000037F'010E0000' 0377     337         .ASCID  /RMS error in file !AD/
20 65 6C 69 66 20 6E 69 20 72 6F 72  0385
                           44 41 21  0391
                                     0394     338
                                     0394     339 DROP_UNIT_MSG:                  ; Follows above msg if testing started
65 63 69 76 65 44 0000039C'010E0000' 0394     340         .ASCID  /Device !AS dropped from testing./
64 65 70 70 6F 72 64 20 53 41 21 20  03A2
6E 69 74 73 65 74 20 6D 6F 72 66 20  03AE
                              2E 67  03BA
```

```
                                  03BC    341
                                  03BC    342 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43  03BC    343         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65  03C8
                                  20  03D4
                            00000019  03D5    344         PMTSIZ = .-PROMPT
                                  03D5    345
                                  03D5    346 ; List of buffer sizes to use (in bytes, max= 32768).
                                  03D5    347 BUF_SZ_LIST:
                            00000200  03D5    348         .LONG   512
                            000001FF  03D9    349         .LONG   511
                            00000012  03DD    350         .LONG   18
                            00000800  03E1    351         .LONG   2048
                            00008000  03E5    352         .LONG   32768
                            00000005  03E9    353         FILE_SZ=.-BUF_SZ_LIST/4
                            00000000  03E9    354         .LONG   0                       ; terminator
                                  03ED    355
                                  03ED    356 ; Densities- (The length of all entries must be equal to DENS_LEN and
                                  03ED    357 ;               end with a space)
                                  03ED    358
                                  03ED    359 DENS_LIST:
                      20 30 30 38 20  03ED    360 NRZI:   .ASCII  / 800 /
                      20 30 30 36 31  03F2    361 PE:     .ASCII  /1600 /
                      20 30 35 32 36  03F7    362 GCR:    .ASCII  /6250 /
                            00000000  03FC    363         .LONG   0                       ; terminator
                                  0400    364
                                  0400    365 ; $GETJPI to get the base priority of the parent process
                                  0400    366
                                  0400    367 GET_LIS:
                                0004  0400    368         .WORD 4
                                0309  0402    369         .WORD JPI$_PRIB
                           00000193'  0404    370         .ADDRESS BASPRI
                            00000000  0408    371         .LONG 0
                            00000000  040C    372         .LONG 0
                                  0410    373
                                  0410    374 ; The following data is used for creating and running indirect commands.
                                  0410    375
                                  0410    376 LOGINOUT:
59 53 24 53 59 53 00000418'010E0000'  0410    377         .ASCID  /SYS$SYSTEM:LOGINOUT.EXE/
55 4F 4E 49 47 4F 4C 3A 4D 45 54 53  041E
                      45 58 45 2E 54  042A
                                  042F    378
                                  042F    379 CMD_OUT:                                ; Command file output descriptor
                          0000 0003'  042F    380         .WORD OUT_LEN,0
                           00000437'  0433    381         .ADDRESS OUT_DEV
                                  0437    382 OUT_DEV:                                ; Output to null device
                            3A 4C 4E  0437    383         .ASCII /NL:/
                            00000003  043A    384         OUT_LEN = .-OUT_DEV
                                  043A    385
                                  043A    386 CMD_FILE:
50 41 54 47 41 4D 00000442'010E0000'  043A    387         .ASCID  /MAGTAPE.COM/
                      4D 4F 43 2E 45  0448
                                  044D    388
                                  044D    389
                                  044D    390
```

```
                                 044D          .SBTTL  Read/Write Data
                             00000000          .PSECT  RWDATA,WRT,NOEXE,PAGE
                                 0000
                                 0000  TTCHAN:                                 ; Channel associated with ctrl. term.
                           0000  0000          .WORD   0
                                 0002
                                 0002  FLAG:                                   ; Miscellaneous flag bits
                           0000  0002          .WORD   0                       ; (See Equated Symbols for definitions)
                                 0004
                                 0004  FAO_BUF:                                ; FAO output string descriptor
                      0000 0084  0004          .WORD   TEXT_BUFFER,0
                      00000014'  0008          .ADDRESS BUFFER
                                 000C
                                 000C  BUFFER_PTR:                             ; Fake .ASCID buffer for misc. strings
                      0000 0084  000C          .WORD   TEXT_BUFFER,0           ; A word for length, a word for desc.
                      00000014'  0010          .ADDRESS BUFFER
                                 0014
                                 0014  BUFFER:                                 ; FAO output and other misc. buffer
                       00000098  0014          .BLKB   TEXT_BUFFER
                                 0098
                                 0098  CUR_UNTBLK:                             ; Address of current unit block
                       00000000  0098          .LONG   0
                                 009C
                                 009C  DEVDSC:                                 ; Device name descriptor
                      0000 000A  009C          .WORD   MAX_DEV_DESIG,0         ; -This will have actual length of DDcn stri
                      000000F8'  00A0          .ADDRESS DEV_NAME
                                 00A4
                                 00A4  LOGNAM_DESC:                            ; Logical name for first testable device
                      0000000A'  00A4          .LONG   LOGNAM_LEN              ;  found that can be used by other tests
                      000000AC'  00A8          .ADDRESS LOGNAM
                                 00AC
  50 41 54 47 41 4D 24 54 45 55  00AC  LOGNAM: .ASCII  /UET$MAGTAP/
                       0000000A  00B6         LOGNAM_LEN=.-LOGNAM
                                 00B6
                                 00B6  ONESHOT_DESC:                           ; File name descriptor - used for tape
                     0000 000A'  00B6          .WORD   ONESHOT_LEN,0           ; ... record in oneshot mode
                      000000BE'  00BA          .ADDRESS OS_FILNM
                                 00BE
                                 00BE  OS_FILNM:
  31 3B 54 41 44 2E 50 54 45 55  00BE          .ASCII /UETP.DAT;1/             ; File version num. required for QIO's
                       0000000A  00C8         ONESHOT_LEN=.-OS_FILNM
                                 00C8
                                 00C8  FILNM_DESC:                             ; File name for tape records - normal
                     0000 0009'  00C8          .WORD   FILNM_LEN,0             ;  and loop mode
                      000000D0'  00CC          .ADDRESS FILNM
                                 00D0
                                 00D0  FILNM:
     54 41 44 2E 50 54 45 55 3A  00D0          .ASCII  /:UETP.DAT/
                       00000009  00D9         FILNM_LEN=.-FILNM
                                 00D9
                                 00D9  TIME:                                   ; Pass duration
              FFFFFFFF 94B62E00  00D9          .LONG   -10*1000*1000*180,-1    ; three minutes to start with
                                 00E1
                                 00E1  PROCESS_NAME:                           ; Process name
  45 50 41 54 000000E9'010E0000' 00E1          .ASCID  /TAPE/
                       0000000B  00ED         PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                       000000F8  00ED          .BLKB   PROCESS_NAME_FREE
```

```
                   00F8
                   00F8  DEV_NAME:                                  ; Device name buffer
          00000107 00F8          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
          0000000F 0107          NAME_LEN = .-DEV_NAME
                   0107
                   0107  DIB:                                       ; Device Information Block
         0000 0074 0107          .WORD   DIB$K_LENGTH,0
         0000010F' 010B          .ADDRESS DIBBUF
                   010F
                   010F  DIBBUF:
          00000183 010F          .BLKB   DIB$K_LENGTH
                   0183
                   0183  ERROR_COUNT:                               ; Cumulative error count at runtime
          00000000 0183          .LONG   0
                   0187
                   0187  STATUS:                                    ; Status value on program exit
          00000000 0187          .LONG   0
                   018B
                   018B  IOSTAT:                                    ; IO status block
 00000000 00000000 018B          .QUAD   0
                   0193
                   0193  BASPRI:                                    ; Base priority received from $Getjpi
          00000002 0193          .LONG   2
                   0197
                   0197  AST_MODE:                                  ; Prior setting of AST delivery
          00000000 0197          .LONG   0
                   019B
                   019B  SS_FAIL_MODE:                              ; Prior setting of SS failure mode
          00000000 019B          .LONG   0
                   019F
                   019F  INADDRESS:                                 ; $CRMPSC address storage
 00000000 00000000 019F          .LONG   0,0
                   01A7
                   01A7  OUTADDRESS:
 00000000 00000000 01A7          .LONG   0,0
                   01AF
                   01AF  UNIT_CNT:                                  ; Number of units found
                00 01AF          .BYTE   0
                   01B0
                   01B0  DEVNAM_LEN:                                ; Current device name length
              0000 01B0          .WORD   0
                   01B2
                   01B2  RANDOM1:                                   ; Used for generating random data
          AAAAAAAA 01B2          .LONG   ^XAAAAAAAA
                   01B6  RANDOM2:
          A72EA72E 01B6          .LONG   ^XA72EA72E
                   01BA
                   01BA  ITERATION:                                 ; Count of the number of files created
          00000000 01BA          .LONG   0                          ; ...in normal and loop modes
                   01BE
                   01BE  PASS:                                      ; Pass count (loop mode)
          00000000 01BE          .LONG   0
                   01C2
                   01C2  MSG_BLOCK:                                 ; Auxiliary $GETMSG info
          000001C6 01C2          .BLKB   4
                   01C6
                   01C6  START_CNT:                                 ; Number of units running
```

```
                          00  01C6    506         .BYTE   0
                              01C7    507
                              01C7    508 EXIT_DESC:                             ; Exit handler descriptor
                    00000000  01C7    509         .LONG   0
                    000012F8' 01CB    510         .ADDRESS EXIT_HANDLER
                    00000001  01CF    511         .LONG   1
                    0000C187' 01D3    512         .ADDRESS STATUS
                              01D7    513
                              01D7    514 ARG_COUNT:                             ; Argument counter used by ERROR_EXIT
                    00000000  01D7    515         .LONG   0
                              01DB    516
                              01DB    517 RMSRUNDWN_BUF:                         ; Return buffer for SYS$RMSRUNDWN close
                   0000 0016  01DB    518         .WORD   22,0                   ;  failures
                    000001E3' 01DF    519         .ADDRESS RUNDWN_BUF
                              01E3    520
                              01E3    521 RUNDWN_BUF:
                    000001F9  01E3    522         .BLKB 22
                              01F9    523
                              01F9    524 ; Head of self-relative UETP unit block queue.
                              01F9    525
                              01F9    526         .ALIGN QUAD
                              0200    527
                              0200    528 UNIT_LIST:                             ; Head of unit block circular list
           00000000 00000000  0200    529         .QUAD   0
                              0208    530
                              0208    531 NEW_NODE:                              ; Newly acquired node address
           00000000 00000000  0208    532         .QUAD   0
                              0210    533
                              0210    534 ; Shared write buffer address
                              0210    535
                              0210    536 WRITE_BUF:
           00000000 00000000  0210    537         .QUAD   0                      ; $EXPREG gets beginning and ending address
                              0218    538
                              0218    539 ; List of buffer start addresses.
                              0218    540
                              0218    541 BUF_ADR_LIST:
                    0000022C  0218    542         .BLKL   FILE_SZ
                              022C    543
                              022C    544 ; The following is used for the INITIALIZE command file creation
                              022C    545
                              022C    546 LABEL_CMD:
           50 54 45 55 20 3A  022C    547         .ASCII  \: UETP\                        ; ...label
                    00000006  0232    548         LABEL_LEN=.-LABEL_CMD
                              0232    549
                              0232    550 CMD_BUF:
3D 53 4E 45 44 2F 54 49 4E 49 24  0232    551         .ASCII  \$INIT/DENS=\                   ; Command
                    0000000B  023D    552         INIT_LEN=.-CMD_BUF
                    00000242  023D    553         .BLKB   DENS_LEN                        ; ...density
                    00000257  0242    554         .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG+LABEL_LEN ; ...unit
                              0257    555
                              0257    556 ; The following is used for subprocess termination mail box - INIT_TAPE
                              0257    557
                              0257    558 MBX_BUF:
                    00000357  0257    559         .BLKB   MBX_SIZE
                              0357    560
                              0357    561 MBX_CHAN:
                        0000  0357    562         .WORD   0
```

```
                     0359
                     0359     MBX_UNIT:
                0000 0359             .WORD   0
                     035B
                     035B     ; define a FIB for oneshot mode (QIO)
                     035B
                     035B     FIB_DESC:
           0000001C' 035B             .LONG   FIB_LEN
           00000363' 035F             .ADDRESS FIB
                     0363
            00000101 0363     FIB:    .LONG   FIB$M_WRITE!FIB$M_NOWRITE ; Read/write access allowed
      0000 0000 0000 0367             .WORD   0,0,0                     ; File ID
      0000 0000 0000 036D             .WORD   0,0,0                     ; Directory ID
            00000000 0373             .LONG   0                         ; Context
                0000 0377             .WORD   0                         ; Name flags
                0000 0379             .WORD   0                         ; Extend control
            00000000 037B             .LONG   0                         ; Control value
            0000001C 037F             FIB_LEN=.-FIB
                     037F
                     037F     MNT_LIST:       ; Item list for mount system service
                000F 037F             .WORD   MAX_DEV_DESIG+MAX_UNIT_DESIG    ; Device name length
                0001 0381             .WORD   MNT$_DEVNAM               ; Item code
           000000F8' 0383             .ADDRESS DEV_NAME                 ; Device name buffer
            00000000 0387             .LONG   0                         ; Unused
                0004 038B             .WORD   4
                0004 038D             .WORD   MNT$_FLAGS                ; Item code
           0000039B' 038F             .ADDRESS MNT_FLAGS                ; Mount flags buffer
            00000000 0393             .LONG   0                         ; Unused
            00000000 0397             .LONG   0                         ; List terminator
                     039B
                     039B     MNT_FLAGS:                                ; Mount flags
            00000204 039B             .LONG   <<MNT$M_NOASSIST>!<MNT$M_OVR_IDENT>>
                     039F
                     039F
```

```
                 039F              .SBTTL  RMS-32 Data Structures
                 039F
                 039F              .ALIGN  LONG
                 03A0
                 03A0      SYSIN_FAB:                              ; Allocate FAB for SYS$INPUT
                 03A0              $FAB-
                 03A0              FNM=<SYS$INPUT>
                 03F0
                 03F0      SYSIN_RAB:                              ; Allocate RAB for SYS$INPUT
                 03F0              $RAB-
                 03F0              FAB=SYSIN_FAB,-
                 03F0              ROP=PMT,-
                 03F0              PBF=PROMPT,-
                 03F0              PSZ=PMTSIZ,-
                 03F0              UBF=DEV_NAME,-
                 03F0              USZ=NAME_LEN
                 0434
                 0434      INI_FAB:                                ; Allocate FAB for UETINIDEV
                 0434              $FAB-
                 0434              FAC = <GET,PUT,UPD>,-
                 0434              RAT = CR,-
                 0434              SHR = <GET,PUT,UPI>,-
                 0434              FNM = <UETINIDEV.DAT>
                 0484
                 0484      INI_RAB:                                ; Allocate RAB for UETINIDEV
                 0484              $RAB-
                 0484              FAB = INI_FAB,-
                 0484              RBF = BUFFER,-
                 0484              UBF = BUFFER,-
                 0484              USZ = REC_SIZE
                 04C8
                 04C8      DDB_RFA:                                ; RFA storage for INI_RAB
        000004CE 04C8              .BLKB   6
                 04CE
                 04CE              .ALIGN  LONG
                 04D0      SUP_FAB:                                ; Allocate FAB for UETSUPDEV
                 04D0              $FAB-
                 04D0              FAC = GET,-
                 04D0              SHR = <UPI,GET>,-
                 04D0              RAT = CR,-
                 04D0              FOP = UFO,-
                 04D0              FNM = <UETSUPDEV.DAT>
                 0520
                 0520      ; Dummy FAB and RAB to copy to the UETP unit blocks
                 0520      ; The following FAB and RAB must be contiguous and in this order!
                 0520
                 0520      DUMMY_FAB:
                 0520              $FAB-
                 0520                      BLS = 512,-
                 0520                      FAC = <BRO,GET,PUT>,-
                 0520                      ORG = SEQ,-
                 0520                      RFM = VAR
                 0570
                 0570      DUMMY_RAB:
                 0570              $RAB-
                 0570                      ROP = <ASY,BIO>,-
                 0570                      USZ = READ_SIZE
```

```
05B4
05B4               .ALIGN  LONG
05B4  CMD_FAB:                                   ; Command file FAB for INIT_TAPE
05B4               $FAB-
05B4                       FNM = <MAGTAPE.COM>,-
05B4                       FAC = PUT,-
05B4                       RAT = CR
0604  INIT_RAB:                                  ; Initialize command RAB
0604               $RAB-
0604                       FAB = CMD_FAB,-
0604                       RBF = CMD_BUF
```

```
                          0648   667           .SBTTL  Main Program
                      00000000   668           .PSECT  TAPE,EXE,NOWRT,PAGE
                          0000   669
                          0000   670           .DEFAULT DISPLACEMENT,WORD
                          0000   671
                          0000   672 ;+
                          0000   673 ;         Start up the tape test.  This entails some overhead necessary to cope
                          0000   674 ;         with both expected and unforseen conditions, figuring out just what
                          0000   675 ;         devices are to be tested, making sure we can test the indicated devices
                          0000   676 ;         and setting up writeable space for each device to be tested.
                          0000   677 ;-
                          0000   678
                    0000  0000   679 .ENTRY UETTAPE00,^M<>                      ; Entry mask
                          0002   680
       6D    1079'CF   DE 0002   681           MOVAL   SSERROR,(FP)              ; Declare exception handler
                          0007   682           $SETSFM_S ENBFLG = #1             ; Enable system service failure mode
                          0010   683           $DCLEXH_S DESBLK = EXIT_DESC      ; Declare an exit handler
                          001B   684
                          001B   685           $OPEN   FAB = SYSIN_FAB,-         ; Open SYS$INPUT
                          001B   686                   ERR = RMS_ERROR
                          002A   687           $CONNECT RAB = SYSIN_RAB,-        ; Connect RAB to SYS$INPUT
                          002A   688                    ERR = RMS_ERROR
                  02   E1 0039   689           BBC     S^#DEV$V_TRM,-            ; BR if SYS$INPUT is NOT a terminal
         1E 03E0'CF       003B   690                   SYSIN_FAB+FAB$L_DEV,10$
                          003F   691           $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                          003F   692                     RSLLEN = DEVNAM_LEN,- ; ...a logial name...
                          003F   693                     RSLBUF = DEVDSC       ; ...for the controller to test
            01    50   D1 0058   694           CMPL    R0,#SS$_NORMAL            ; Was a controller specified?
                  2E   13 005B   695           BEQL    PROC_CONT_NAME            ; BR if it was - go process it
                          005D   696 10$:
                          005D   697           $GET    RAB = SYSIN_RAB,-         ; Read SYS$INPUT...
                          005D   698                   ERR = RMS_ERROR           ; ...for the controller name
           0412'CF     B0 006C   699           MOVW    SYSIN_RAB+RAB$W_RSZ,-     ; Save the name length
           01B0'CF        0070   700                   DEVNAM_LEN
                  16   12 0073   701           BNEQ    PROC_CONT_NAME            ; BR if we got something
       0187'CF    14   D0 0075   702           MOVL    #SS$_BADPARAM,STATUS      ; Save an exit status if not
           00D8'CF     DF 007A   703           PUSHAL  NO_CTRLNAME               ; Prepare for message...
                  01   DD 007E   704           PUSHL   #1                        ; ...arg count
        00741132 8F    DD 0080   705           PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name
                  03   DD 0086   706           PUSHL   #3                        ; ...arg count
                11ED   31 0088   707           BRW     ERROR_EXIT                ; ...go tell of bad setup
                          008B   708
                          008B   709 PROC_CONT_NAME:
   009C'CF     01B0'CF 3C 008B   710           MOVZWL  DEVNAM_LEN,DEVDSC         ; Set the device name length
           009C'CF     DF 0092   711           PUSHAL  DEVDSC                    ; Make sure...
           009C'CF     DF 0096   712           PUSHAL  DEVDSC                    ; ...that the specified controller...
    00000000'GF   02   FB 009A   713           CALLS   #2,G^STR$UPCASE           ; ...is all uppercase for later comaparison
  52   009C'CF    01   C1 00A1   714           ADDL3   #1,DEVDSC,R2              ; Estimate the eventual...
       00E1'CF    52   A0 00A7   715           ADDW2   R2,PROCESS_NAME           ; ...process name length (incl. "_")
                       DE 00AC   716           MOVAL   PROCESS_NAME+8-           ; Locate first available byte...
                          00AD   717                   +MAX_PROC_NAME-           ; ...in process name handle...
       50    00ED'CF      00AD   718                   -PROCESS_NAME_FREE,R0     ; ...for device name
                  0B   C3 00B1   719           SUBL3   #PROCESS_NAME_FREE,-      ; Will the device name fit...
            51    52      00B3   720                   R2,R1                     ; ...in the remaining space?
                  08   15 00B5   721           BLEQ    10$                       ; BR if it will
            50    51   C2 00B7   722           SUBL2   R1,R0                        ; Overwrite handle otherwise...
       00E1'CF    0F   B0 00BA   723           MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
```

```
                                00BF    724 10$:
          80   5F 8F      90    00BF    725         MOVB    #^A/_/,(R0)+              ; Separate handle from device name
 60   00F8'CF   009C'CF   28    00C3    726         MOVC3   DEVDSC,DEV_NAME,(R0)      ; Concatenate handle with device name
                     7E   D4    00CB    727         CLRL    -(SP)                     ; Set the time stamp flag
               000F'CF    DF    00CD    728         PUSHAL  TEST_NAME                 ; Set the test name
                     02   DD    00D1    729         PUSHL   #2                        ; Push the argument count
          00741039 8F     DD    00D3    730         PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
       00000000'GF   04   FB    00D9    731         CALLS   #4,G^LIB$SIGNAL           ; Print the startup message
          0002'CF    08   A8    00E0    732         BISW2   #BEGIN_MSGM,FLAG          ; Set flag so we don't print it again
                                00E5    733         $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETTAPE00_x
                                00F0    734
                     02   E1    00F0    735         BBC     S^#DEV$V_TRM,-            ; BR if SYS$INPUT is NOT a terminal
             66 03E0'CF         00F2    736                 SYSIN_FAB+FAB$L_DEV,20$
                                00F6    737         $GETDVI_S DEVNAM = SYS$INPUT,-   ; Get the name of...
                                00F6    738                   EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                                00F6    739                   ITMLST = INPUT_ITMLST,-
                                00F6    740                   IOSB   = IOSTAT
             45 018B'CF   E9    0112    741         BLBC    IOSTAT,20$                ; Avoid CTRL/C handler if any error
                                0117    742         $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                                0117    743                   CHAN   = TTCHAN
                                0128    744         $QIOW_S CHAN     = TTCHAN,-       ; Enable CTRL/C AST's...
                                0128    745                 FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                                0128    746                 P1       = CCASTHAND
               00E1'CF    DF    0149    747         PUSHAL  PROCESS_NAME              ; ...and tell the user...
                     01   DD    014D    748         PUSHL   #1                        ; ...
          0074832B 8F     DD    014F    749         PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
       00000000'GF   03   FB    0155    750         CALLS   #3,G^LIB$SIGNAL           ; ...
                                015C    751 20$:
```

```
                               015C
                               015C      ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                               015C      ; and unit configuration and lets us know if the setup to run this test was
                               015C      ; done correctly.
                               015C
                               015C              $OPEN    FAB = INI_FAB,-            ; Open file 'UETINIDEV.DAT''
                               015C                       ERR = RMS_ERROR
                               016B              $CONNECT RAB = INI_RAB,-            ; Connect the RAB and FAB
                               016B                         ERR = RMS_ERROR
                               017A              $MGBLSC_S  INADR = INADDRESS,-  ; Connect to UETSUPDEV global section -
                               017A                           RETADR = OUTADDRESS,- ;   if it is there
                               017A                          GSDNAM = SUPDEV_GBLSEC,-
                               017A                            FLAGS = #SEC$M_EXPREG
        00000978 8F    50  D1  0199              CMPL     R0,#SS$_NOSUCHSEC          ; Was the section already there?
                       37  12  01A0              BNEQ     30$                        ; BR if it was...
                               01A2              $OPEN    FAB = SUP_FAB,-            ; ...else open 'UETSUPDEV.DAT''
                               01A2                       ERR = RMS_ERROR
                               01B1              $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                               01B1                         INADR = INADDRESS,-
                               01B1                       RETADR = OUTADDRESS,-
                               01B1                       GSDNAM = SUPDEV_GBLSEC,-
                               01B1                         FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                               01D9
                               01D9     30$:     ; We have a global section
                               01D9
     59   01AB'CF    01A7'CF   C3  01D9              SUBL3    OUTADDRESS,OUTADDRESS+4,R9 ; Compute global section length
                               01E1
                               01E1     FIND_IT: ; Let's look for a DDB
                               01E1
                               01E1              $GET     RAB = INI_RAB,-            ; Get the first record
                               01E1                       ERR = RMS_ERROR
                036F'CF    DF  01F0              PUSHAL   CONT_DESC                  ; Make sure...
                036F'CF    DF  01F4              PUSHAL   CONT_DESC                  ; ...that the controller name...
            00000000'GF    02  FB  01F8          CALLS    #2,G^STR$UPCASE            ; ...is all uppercase letters
             0014'CF    44 8F  91  01FF          CMPB     #^A/D/,BUFFER              ; Is this a DDB?
                           27  13  0205          BEQL     10$                        ; BR if it is
             0014'CF    45 8F  91  0207          CMPB     #^A/E/,BUFFER              ; Is this the end of the file?
                           D2  12  020D          BNEQ     FIND_IT                    ; If not - look again
                009C'CF    DF  020F              PUSHAL   DEVDSC                     ; We are at EOF and a matching DDB was
                00E1'CF    DF  0213              PUSHAL   PROCESS_NAME               ; not found, bitch about it and quit
                     02    DD  0217              PUSHL    #2                         ; ...arg count
            00748333 8F    DD  0219              PUSHL    #UETP$_DENOSU              ; ...signal name
                     02    F0  021F              INSV     #STS$K_ERROR,-             ; Set the severity code...
                     00        0221                       #STS$V_SEVERITY,-
                  6E 03        0222                       #STS$S_SEVERITY,(SP)
             0187'CF    6E  D0  0224             MOVL     (SP),STATUS                ; ...and save it as the exit status
                     04    DD  0229              PUSHL    #4                         ; ...arg count
                   104A    31  022B              BRW      ERROR_EXIT                 ; Exit in error
                               022E
                               022E     10$:     ; We found a DDB
                               022E
  00F8'CF   001A'CF    01B0'CF  29  022E          CMPC     DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                           A7  12  0238          BNEQ     FIND_IT                    ; If not, look some more
       04C8'CF    0494'CF  06  28  023A          MOVC3    #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
             0018'CF    54 8F  91  0242          CMPB     #^A/T/,BUFFER+4            ; Is controller marked testable?
                           2F  13  0248          BEQL     FOUND_IT                   ; BR if it is testable
                               024A              $FAO_S   CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if it isn't
```

```
                                    024A                      OUTLEN = BUFFER_PTR,-
                                    024A                      OUTBUF = FAO_BUF,-
                                    024A                      P1     = #DEVDSC         ; Bad controller designation
                  0187'CF   14  D0  0263            MOVL      #SS$_BADPARAM,STATUS     ; Set return status
                       000C'CF  DF  0268            PUSHAL    BUFFER_PTR               ; ...
                            01  DD  026C            PUSHL     #1                       ; ...
                   00741132 8F  DD  026E            PUSHL     #UETP$_TEXT!STS$K_ERROR  ; ...
                            03  DD  0274            PUSHL     #3                       ; ...
                          0FFF  31  0276            BRW       ERROR_EXIT               ; We can't test what we can't test
                                    0279
                                    0279  FOUND_IT: ; We have the right controller - let's look for UCB's
                                    0279
                                    0279            $GET      RAB = INI_RAB,-          ; Get a record
                                    0279                      ERR = RMS_ERROR
                       036F'CF  DF  0288            PUSHAL    CONT_DESC                ; Make sure...
                       036F'CF  DF  028C            PUSHAL    CONT_DESC                ; ...that this line...
             00000000'GF    02  FB  0290            CALLS     #2,G^STR$UPCASE          ; ...is all uppercase letters
              0014'CF    55 8F  91  0297            CMPB      #^A/U/,BUFFER            ; Is this a UCB?
                            24  13  029D            BEQL      30$                      ; BR if it is
              0014'CF    44 8F  91  029F            CMPB      #^A/D/,BUFFER            ; Is this a DDB?
                            19  13  02A5            BEQL      20$                      ; BR if yes
              0014'CF    45 8F  91  02A7            CMPB      #^A/E/,BUFFER            ; Is this the end?
                            11  13  02AD            BEQL      20$                      ; BR if yes
                                    02AF
                                    02AF  10$:      ; Something is wrong with the contents of UETINIDEV.DAT
                                    02AF
                       017F'CF  DF  02AF            PUSHAL    ILLEGAL_REC              ; Then this is an error in the record
                            01  DD  02B3            PUSHL     #1                       ; Push the error message
                   00741132 8F  DD  02B5            PUSHL     #UETP$_TEXT!STS$K_ERROR  ; Push the signal name
                            03  DD  02BB            PUSHL     #3                       ; Push the temp arg count
                          0FB8  31  02BD            BRW       ERROR_EXIT               ; Finish for good
                                    02C0
                                    02C0  20$:      ; Found another DDB or END
                                    02C0
                          0138  31  02C0            BRW       ALL_SET
                                    02C3
                                    02C3  30$:      ; We found a UCB
                                    02C3
              0018'CF    54 8F  91  02C3            CMPB      #^A/T/,BUFFER+4          ; Is the unit testable?
                            AE  12  02C9            BNEQ      FOUND_IT                 ; If not, look some more
                      05    20  3B  02CB            SKPC      #^A/ /,#MAX_UNIT_DESIG,- ; Find out where unit number really is
                       001A'CF      02CE                      BUFFER+6
                            50  D7  02D1            DECL      R0                       ; Units must all be at least one digit
                61    50    30  3B  02D3            SKPC      #^A/0/,R0,(R1)           ; Skip leading zeroes on the unit
                            50  D6  02D7            INCL      R0                       ; Compensate for DECL above
       009C'CF   01B0'CF    50  A1  02D9            ADDW3     R0,DEVNAM_LEN,DEVDSC     ; Calculate device'unit string length
                  52   01B0'CF  3C  02E1            MOVZWL    DEVNAM_LEN,R2            ; Offset to unit number in DEVDSC
           00F8'C2    61    50  28  02E6            MOVC3     R0,(R1),DEV_NAME(R2)     ; Append unit number to device
                                    02EC            $GETDEV_S DEVNAM = DEVDSC,-        ; Get the device characteristics
                                    02EC                        PRIBUF = DIB
                  57   0113'CF  9A  0301            MOVZBL    DIBBUF+DIB$B_DEVCLASS,R7 ; Save the device class
                  58   0114'CF  9A  0306            MOVZBL    DIBBUF+DIB$B_DEVTYPE,R8  ; Save the device type
                                    030B            $FAO_S    CTRSTR = CS1,-           ; Make it into a string
                                    030B                      OUTBUF = FAO_BUF,-
                                    030B                      P1     = R7,-
                                    030B                      P2     = R8
 01A7'DF    59   0014'CF    06  39  0320            MATCHC    #6,BUFFER,R9,@OUTADDRESS ; Find the device class and type
```

```
                         1E   13 0329           BEQL    40$                     ; BR if it was found
                                 032B           $FAO_S  CTRSTR = CS3,-          ; Try for full class support
                                 032B                   OUTBUF = FAO_BUF,-
                                 032B                       P1 = R7
 01A7'DF   59   0014'CF  06   39 033E           MATCHC  #6,BUFFER,R9,@OUTADDRESS ; Find the device class only
                         0D   12 0347           BNEQ    50$                     ; BR if not found
                                 0349
                                 0349     40$:  ; Device type and class are correct - what about test?
                                 0349
                55    000F'CF 9A 0349           MOVZBL  TEST_NAME,R5            ; Get the test name length
       0017'CF  63       55   29 034E           CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
                         1F   13 0354           BEQL    60$                     ; BR if yes
                                 0356
                                 0356     50$:  ; Can't make heads or tails out of this device - bitch and quit
                                 0356
                    009C'CF   DF 0356           PUSHAL  DEVDSC                  ; Push device not supported message
                    00E1'CF   DF 035A           PUSHAL  PROCESS_NAME            ; Parameters on the stack
                         02   DD 035E           PUSHL   #2                      ; Push the argument count
                00748333 8F   DD 0360           PUSHL   #UETP$_DENOSU
                         02   F0 0366           INSV    #STS$K_ERROR,-
                         00      0368                   #STS$V_SEVERITY,-
                    6E   03      0369                   #STS$S_SEVERITY,(SP)    ; Set the severity code...
              0187'CF    6E   D0 036B           MOVL    (SP),STATUS             ; ...and save it as the exit status
                         04   DD 0370           PUSHL   #4                      ; Push the partial arg count...
                       0F03   31 0372           BRW     ERROR_EXIT              ; ...and split this scene
                                 0375
```

```
0375  ;+
0375  ; The following code dynamically allocates enough memory for a unit block,
0375  ; a device dependent parameter area and I/O buffers. The unit block is inserted
0375  ; into the queue header UNIT_LIST.  It then initializes the unit block.
0375  ; A comment indicates where the device dependent parameters should be
0375  ; initialized.  The unit block format is as follows:
0375  ;
0375  ;                        +---------------+ -------
0375  ;       UETUNT$L_FLINK   !               !       ^
0375  ;                        +---------------+       !
0375  ;       UETUNT$L_BLINK   !               !       !
0375  ;                        +-----------+---+       !
0375  ;       UETUNT$B_TYPE                !   !       !
0375  ;                            +-------+---+       !
0375  ;       UETUNT$W_SIZE        !   !   !    contains DEVDEP_SIZE + UETUNT$C_INDSIZ
0375  ;                        +---+-------+           !
0375  ;       UETUNT$B_FLAGS   !   !                   !
0375  ;                        +---+   +-------+       !
0375  ;       UETUNT$W_CHAN            !   !   !       !
0375  ;                        +-------+-------+       !
0375  ;       UETUNT$W_FUNC    !   !   !               !
0375  ;                        +-------+-------+       +----- UETUNT$C_SIZE
0375  ;       UETUNT$L_ITER    !               !       !
0375  ;                        +---------------+
0375  ;       UETUNT$T_FILSPC  !               !
0375  ;                        !/\/\/\/\/\/\/\/!
0375  ;                        NAM$C_MAXRSS bytes      .
0375  ;                        !/\/\7\/\/\/\/\/!
0375  ;                        !               !
0375  ;       UETUNT$K_FAB     !---------------.
0375  ;                        !               !
0375  ;                        !/\/\/\/\/\/\/\/!       !
0375  ;                         FAB$C_BLN bytes        !
0375  ;                        !/\/\/\/\/\/\/\/.       !
0375  ;                        !               !       .
0375  ;       UETUNT$K_RAB     +---------------+       .
0375  ;                        !               !       !
0375  ;                        !\/\/\/\/\/\/\/\!       !
0375  ;                         RAB$C_BLN bytes        !
0375  ;                        !\/\/\7\/\/\/\/\!       !
0375  ;                        !               !       v
0375  ;       UETUNT$K_DEVDEP  +---------------+ -------
0375  ;       UETUNTK$_DENSITY!               !       ^
0375  ;                        !/\/\/\/\/\/\/\/!       !
0375  ;                            DENS_LEN            !
0375  ;                        !/\/\/\/\/\/\/\/!       !
0375  ;                        !               !       !
0375  ;                        +---------------+       !
0375  ;       UETUNT$Q_DEVDSC  !               !       !
0375  ;                        +---------------+       !
0375  ;                        !               !       !
0375  ;                        +---------------+       +-----DEVDEP_SIZE
0375  ;       UETUNT$K_DEV_NAM!               !       !
0375  ;                        !\/\/\/\/\/\/\/\!       !
0375  ;                          MAX_DEV_DESIG+        !
0375  ;                          MAX_UNIT_DESIG        !
0375  ;                        !/\/\7\/\/\/\/\/!       !
```

```
                                0375   ;                       !               !       !
                                0375   ;                       +-----------+---+       !
                                0375   ;       UETUNT$B_BUFPTR             !   !       !
                                0375   ;                               +---+---+       !
                                0375   ;       UETUNT$B_DENSPTR        !   !           v
                                0375   ;                       +-------+---+  ----------
                                0375   ;       UETUNT$K_RBUF   !       !               ^
                                0375   ;                       !       +-------+       !
                                0375   ;                       !\/\/\/\/\/\/\/\!       +----- READ_SIZE
                                0375   ;       READ buffer         READ_SIZE           !
                                0375   ;                       !/\/\/\/\/\/\/\/!       !
                                0375   ;                       !               !       v
                                0375   ;                       +---------------+ -------
                                0375   ;-
                                0375
                                0375   60$:    $EXPREG_S PAGCNT = #PAGES,-
                                0375                     RETADR = NEW_NODE        ; Get a new node of demand zero memory
0200'CF   0208'DF   5D   038A          INSQTI  @NEW_NODE,UNIT_LIST        ; Put the new node in the unit list
     56   0208'CF   D0   0391          MOVL    NEW_NODE,R6                ; Save a copy of its address
       08 A6   01   90   0396          MOVB    #1,UETUNT$B_TYPE(R6)       ; Set the structure type
          01C2 8F   B0   039A          MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
            09 A6        039E                  UETUNT$W_SIZE(R6)          ; Set the structure size
     009C'CF   09   81   03A0          ADDB3   #FILNM_LEN,DEVDSC,-
            14 A6        03A5                  UETUNT$T_FILSPC(R6)        ; Set the device name size
00A0'DF   009C'CF   28   03A7          MOVC3   DEVDSC,@DEVDSC+4,-
            15 A6        03AE                  UETUNT$T_FILSPC+1(R6)      ; Save the device name
63   00D0'CF   09   28   03B0          MOVC3   #FILNM_LEN,FILNM,(R3)      ; Rest of name
          0094 8F   28   03B6          MOVC3   #FAB$C_BLN+RAB$C_BLN,-
0110 C6   0520'CF        03BA                  DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
     57   0110 C6   DE   03C0          MOVAL   UETUNT$K_FAB(R6),R7        ; Save the FAB address
     58   0160 C6   DE   03C5          MOVAL   UETUNT$K_RAB(R6),R8        ; Save the RAB address
       3C A8   57   D0   03CA          MOVL    R7,RAB$L_FAB(R8)           ; Set the FAB address in the RAB
            14 A6   90   03CE          MOVB    UETUNT$T_FILSPC(R6),-
            34 A7        03D1                  FAB$B_FNS(R7)              ; Set the FNS field in the FAB
            15 A6   DE   03D3          MOVAL   UETUNT$T_FILSPC+1(R6),-
            2C A7        03D6                  FAB$L_FNA(R7)              ; Set the FNA field in the FAB
       18 A8   66   DE   03D8          MOVAL   (R6),RAB$L_CTX(R8)         ; Set the UETUNT address in the RAB
       18 A7   66   DE   03DC          MOVAL   (R6),FAB$L_CTX(R7)         ;   and in the FAB
          01B1 C6   DE   03E0          MOVAL   UETUNT$K_DEV_NAM(R6),-     ; Setup addr of device name descriptor
          01AD C6        03E4                  UETUNT$Q_DEVDSC+4(R6)      ;   in the unit block
01A9 C6   009C'CF   D0   03E7          MOVL    DEVDSC,UETUNT$Q_DEVDSC(R6) ; Setup device name length
00F8'CF   009C'CF   28   03EE          MOVC3   DEVDSC,DEV_NAME,-
          01B1 C6        03F5                  UETUNT$K_DEV_NAM(R6)       ; Save the device name
             FE7E   31   03F8          BRW     FOUND_IT                   ; We are doing so well let's look
                         03FB                                             ;   for more UCB's
```

```
                          03FB
                          03FB  ; Arrive here when we have the device configuration.  In normal or loop forever
                          03FB  ; mode, set a timer far enough in the future such that we can do a reasonable
                          03FB  ; set of tests before the timer expires, but if our device gets hung, the
                          03FB  ; program won't waste too much time before noticing.  Let one-shot mode be a
                          03FB  ; special case.
                          03FB
                          03FB  ALL_SET:
         0200'CF    D5    03FB          TSTL    UNIT_LIST               ; Anything to test?
              16    12    03FF          BNEQ    10$                     ; BR if yes
         013F'CF    DF    0401          PUSHAL  NOUNIT_SELECTED         ; Else set up the error message...
              01    DD    0405          PUSHL   #1                      ; ...argument count...
     00741132 8F    DD    0407          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name...
              03    DD    040D          PUSHL   #3                      ; ...and parameter count
     0187'CF    14  D0    040F          MOVL    #SS$_BADPARAM,STATUS    ; Set return status
          0E61      31    0414          BRW     ERROR_EXIT              ; ...and give up, complaining
                          0417  10$:
                          0417          $EXPREG_S-                      ; Get memory for common write buffer
                          0417                  PAGCNT = #WRITE_SIZE+511/512,-
                          0417                  RETADR = WRITE_BUF
                          042C
                          042C  ; Load write buffer with random data.
                          042C
      56    0210'CF   D0  042C          MOVL    WRITE_BUF,R6            ; Get buffer address
  57    00002000 8F   D0  0431          MOVL    #WRITE_SIZE+3/4,R7      ; Longword size
                          0438  20$:
 01B2'CF    01B6'CF   C0  0438          ADDL2   RANDOM2,RANDOM1         ; Get random longword
      86    01B2'CF   D0  043F          MOVL    RANDOM1,(R6)+           ; Save it
             F1 57    F5  0444          SOBGTR  R7,20$                  ; Continue until done
                          0447
                          0447  ; The following code gets buffer starting addresses.
                          0447  ; Buffers start at different places in the common buffer
                          0447  ; to vary the data pattern between records.
                          0447
      57    03D5'CF   DE  0447          MOVAL   BUF_SZ_LIST,R7          ; Address of size list
      58    0218'CF   DE  044C          MOVAL   BUF_ADR_LIST,R8         ; Address of address list
      59    0214'CF   D0  0451          MOVL    WRITE_BUF+4,R9          ; Get end of buffer
                          0456  30$:
      88     59    87  C3 0456          SUBL3   (R7)+,R9,(R8)+          ; Subtract size to get start address
                   67  D5 045A          TSTL    (R7)                    ; End of list?
                   F8  12 045C          BNEQ    30$                     ; If not
                   00  DD 045E          PUSHL   #0                      ; Zero indicates startup (not loop)
     0002'CF       04  A8 0460          BISW2   #SAFE_TO_UPDM,FLAG      ; OK safe to update UETINIDEV.DAT now
     0A81'CF       01  FB 0465          CALLS   #1,MOUNT_TAPE           ; Let's go mount the tape(s)
                          046A          $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                          046A                    RSLLEN = BUFFER_PTR,-
                          046A                    RSLBUF = FAO_BUF
     0014'CF       20  8A 0483          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
   0014'CF     4F  8F  91 0488          CMPB    #^A7O/,BUFFER           ; Is this a one shot?
                   2D  13 048E          BEQL    50$
   0014'CF     4C  8F  91 0490          CMPB    #^A/L/,BUFFER           ; Is this loop?
                   05  12 0496          BNEQ    40$
     0002'CF       20  A8 0498          BISW2   #LOOP_MODM,FLAG         ; Set loop mode
                          049D  40$:
     01AF'CF       01  91 049D          CMPB    #1,UNIT_CNT             ; Is there only one unit to test?
                   21  13 04A2          BEQL    RESTART                 ; If only one unit go ahead and start
     5B    01AF'CF     9A 04A4          MOVZBL  UNIT_CNT,R11            ; Get unit count
```

```
                      5B   D7  04A9   1054           DECL    R11                         ; Subtract first unit
       5A   034F'CF   5B   C5  04AB   1055           MULL3   R11,THIRTYSEC,R10           ; Add thirty seconds of run time for
            5A   0363'CF   C0  04B1   1056           ADDL2   THREEMIN,R10                ; ...each unit after the first
            00D9'CF   5A   CE  04B6   1057           MNEGL   R10,TIME                    ; Compliment for delta time
                      08   11  04BB   1058           BRB     RESTART
                               04BD   1059 50$:
            0002'CF   10   A8  04BD   1060           BISW2   #ONESHOT_MODM,FLAG          ; Set one shot mode flag
                    0301   31  04C2   1061           BRW     ONE_SHOT
                               04C5   1062
```

```
                                      04C5                .SBTTL  Test the Magtape
                                      04C5
                                      04C5        RESTART:    ; Here we start testing in normal and loop modes.
                                      04C5
                                      04C5        ;*************************************************************************
                                      04C5        ;
                                      04C5        ;  This routine starts off each unit by first synchronously creating a tape
                                      04C5        ; file and connecting to it, then it starts an asynchronous WRITE. Once started,
                                      04C5        ; all units are then run asycronously using AST's. WRITEs are continued until
                                      04C5        ; all buffer sizes in the buffer size list are written.  The file is CLOSEd
                                      04C5        ; and SPACEd in reverse to prepare for reading the records.  Then READs are
                                      04C5        ; done to the end of the file with each record data checked.  When all records
                                      04C5        ; have been read a new file is started with a CREATE, and WRITEs are started
                                      04C5        ; again.  This continues until a timeout (three minutes + <the number of units
                                      04C5        ; - 1> * 30 seconds) occurs which was set at the start. After the timeout a
                                      04C5        ; thirty second watch dog timer is set and each unit completes the file it is
                                      04C5        ; processing and does a REWIND. If loop mode is set, the whole routine is
                                      04C5        ; repeated indefinitely.
                                      04C5
                                      04C5                $SETIMR_S DAYTIM = TIME,-       ; Testing will continue until this
                                      04C5                          EFN    = #2,-
                                      04C5                          ASTADR = TIME_OUT,-   ; timer expires
                                      04C5                          REQIDT = #REQIDT1     ; End of pass ID
         0002'CF    0080 8F     A8    04D8                BISW2   #TEST_STARTM,FLAG     ; Testing has started
                    01C6'CF     94    04DF                CLRB    START_CNT             ; Initialize units started counter
   57   0200'CF   00000200'8F   C1    04E3                ADDL3   #UNIT_LIST,UNIT_LIST,R7 ; Set the unit block list header
                                      04ED
                                      04ED        LOOP:   ; We return here until all units are started.
                                      04ED
                    01AF'CF     97    04ED                DECB    UNIT_CNT              ; Decr the number of units to start
                         01     E1    04F1                BBC     #UETUNT$V_TESTABLE,-  ; If unit not testable skip to next one
                   53 0B A7           04F3                        UETUNT$B_FLAGS(R7),20$
                    01C6'CF     96    04F6                INCB    START_CNT             ; Count the units as we start them
                         10     88    04FA                BISB2   #UETUNT$M_MODIFIED,-  ; Let's flag tape as modified before we
                      0B A7           04FC                        UETUNT$B_FLAGS(R7)    ; ...do it in case we get an error
                                      04FE                $CREATE-                      ; Create a file
                                      04FE                        FAB = UETUNT$K_FAB(R7),-
                                      04FE                        ERR = RMS_ERROR
                      39 50     E9    050D                BLBC    R0,20$                ; BR on error
            56      0160 C7     DE    0510                MOVAL   UETUNT$K_RAB(R7),R6   ; Get RAB address
                                      0515                $CONNECT-                     ; Connect RAB
                                      0515                        RAB = (R6),-
                                      0515                        ERR = RMS_ERROR
                      24 50     E9    0522                BLBC    R0,20$                ; BR on error
                    01C0 C7     94    0525                CLRB    UETUNT$B_BUFPTR(R7)   ; Initialize buffer list index
             22 A6  03D5'CF     B0    0529                MOVW    BUF_SZ_LIST,RAB$W_RSZ(R6) ; Set first buffer size in RAB
             28 A6  0218'CF     D0    052F                MOVL    BUF_ADR_LIST,RAB$L_RBF(R6) ; Set first buffer addr in RAB
                                      0535                $WRITE-                       ; Write a record
                                      0535                        RAB = (R6),-
                                      0535                        SUC = AST_WRITE,-
                                      0535                        ERR = RMS_ERROR
                      00 50     E9    0546                BLBC    R0,20$                ; BR on error
                                      0549
                                      0549        20$:    ; Time to start next unit - if there is more
                                      0549
                      57   67   C0    0549                ADDL2   (R7),R7               ; Next unit block
          00000200'8F      57   D1    054C                CMPL    R7,#UNIT_LIST         ; Done all units?
```

```
                     2F   12  0553   1121           BNEQ    60$                     ; If not, start another
               01C6'CF    95  0555   1122           TSTB    START_CNT               ; Any unit started ok?
                     0D   15  0559   1123           BLEQ    40$                     ; BR if none
                              055B   1124           $HIBER_S                        ; All testable units started- wait here
                              0562   1125 30$:
               01C6'CF    95  0562   1126           TSTB    START_CNT               ; Have all units finished?
                     13   14  0566   1127           BGTR    50$                     ; If not branch
                              0568   1128 40$:
                              0568   1129           $CANTIM_S                       ; Cancel pass timer if no units started
                              0571   1130                                           ; sucessfully, otherwise cancel
                              0571   1131                                           ; watch dog timer.
 0002'CF     0080 8F      AA  0571   1132           BICW2   #TEST_STARTM,FLAG       ; We are done testing - clear flag
                   0499   31  0578   1133           BRW     END_PASS                ; Exit the pass
                              057B   1134 50$:
                              057B   1135           $HIBER_S                        ; Wait here for all to finish
                     DE   11  0582   1136           BRB     30$
                   FF66   31  0584   1137 60$:      BRW     LOOP                    ; Go start next unit
                              0587   1138
                              0587   1139 ; Enter here after a WRITE.  Issues the next WRITE unless we are at
                              0587   1140 ; end of buffer list, when it SPACEs back over records to prepare
                              0587   1141 ; for READs.
                              0587   1142
                              0587   1143 AST_WRITE:
                        0FFC  0587   1144           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
           56    04 AC    D0  0589   1145           MOVL    4(AP),R6                ; Get RAB address
           57    18 A6    D0  058D   1146           MOVL    RAB$L_CTX(R6),R7        ; Get unit block address
                     01   E1  0591   1147           BBC     #UETUNT$V_TESTABLE,-    ; If unit not testable quit trying
              47 0B A7        0593   1148                   UETUNT$B_FLAGS(R7),20$
        01C0 C7      04   80  0596   1149           ADDB    #4,UETUNT$B_BUFPTR(R7)  ; Set index for next buffer
        58   01C0 C7      9A  059B   1150           MOVZBL  UETUNT$B_BUFPTR(R7),R8  ; Get buffer list index
               03D5'C8    D5  05A0   1151           TSTL    BUF_SZ_LIST(R8)         ; Is it the terminator?
                     1B   12  05A4   1152           BNEQ    10$                     ; If not
  38 A6    FFFFFFFB 8F    D0  05A6   1153           MOVL    #-FILE_SZ,RAB$L_BKT(R6) ; Set blocks to skip(reverse)
                              05AE   1154           $SPACE-                         ; If done writing, get ready to read
                              05AE   1155                   RAB = (R6),-
                              05AE   1156                   SUC = AST_SPACE,-
                              05AE   1157                   ERR = RMS_ERROR
                     1C   11  05BF   1158           BRB     20$
                              05C1   1159 10$:
     22 A6    03D5'C8     B0  05C1   1160           MOVW    BUF_SZ_LIST(R8),RAB$W_RSZ(R6) ; Set RAB buffer size
     28 A6    0218'C8     D0  05C7   1161           MOVL    BUF_ADR_LIST(R8),RAB$L_RBF(R6) ; Set buffer address
                              05CD   1162           $WRITE-                         ; Write the next record
                              05CD   1163                   RAB = (R6),-
                              05CD   1164                   SUC = AST_WRITE,-
                              05CD   1165                   ERR = RMS_ERROR
                          04  05DD   1166 20$:      RET
                              05DE   1167
                              05DE   1168
                              05DE   1169 ; Entered from a space function.  Starts up a READ on the first record
                              05DE   1170 ; in the file.
                              05DE   1171
                              05DE   1172 AST_SPACE:
                        0FFC  05DE   1173           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
           56    04 AC    D0  05E0   1174           MOVL    4(AP),R6                ; Get RAB address
           57    18 A6    D0  05E4   1175           MOVL    RAB$L_CTX(R6),R7        ; Get unit block address
                     01   E1  05E8   1176           BBC     #UETUNT$V_TESTABLE,-    ; If unit not testable quit trying
              21 0B A7        05EA   1177                   UETUNT$B_FLAGS(R7),10$
```

```
               01C0 C7   94  05ED          CLRB    UETUNT$B_BUFPTR(R7)          ; Initialize buffer list index
         20 A6  03D5'CF  B0  05F1          MOVW    BUF_SZ_LIST,RAB$W_USZ(R6)    ; Use 1st list entry
         24 A6  01C2 C7  DE  05F7          MOVAL   UETUNT$K_RBUF(R7),RAB$L_UBF(R6) ; Read buffer address
                             05FD          $READ-                               ; Read 1st record
                             05FD                  RAB = (R6),-
                             05FD                  SUC = AST_READ,-
                             05FD                  ERR = RMS_ERROR
                         04  060E    10$:  RET
                             060F
                             060F    ; Entered from READ function.  Checks the record just read and starts
                             060F    ; another READ, unless at the end of the buffer size list, when it
                             060F    ; CLOSEs the file.
                             060F
                             060F    AST_READ:
                       0FFC  060F          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
            56    04 AC  D0  0611          MOVL    4(AP),R6                     ; Get RAB address
            57    18 A6  D0  0615          MOVL    RAB$L_CTX(R6),R7             ; Get unit block address
                     01  E1  0619          BBC     #UETUNT$V_TESTABLE,-         ; If unit not testable quit trying
              16 0B A7       061B                  UETUNT$B_FLAGS(R7),05$
            58  01C0 C7  9A  061E          MOVZBL  UETUNT$B_BUFPTR(R7),R8       ; Get buffer list index
            59  03D5'C8  D0  0623          MOVL    BUF_SZ_LIST(R8),R9           ; Get size of last read
            5A  0218'C8  D0  0628          MOVL    BUF_ADR_LIST(R8),R10         ; Get write buffer address
            5B  01C2 C7  DE  062D          MOVAL   UETUNT$K_RBUF(R7),R11        ; Get read buffer address
                     03  11  0632          BRB     10$
                             0634    05$:
                   00D8  31  0634          BRW     40$                          ; Branch byte won't reach
                             0637
                             0637    10$:  ; Compare data read to data writtem
                             0637
                 8B    8A  91  0637        CMPB    (R10)+,(R11)+                ; Check the byte
                       06  12  063A        BNEQ    20$                          ; BR if mismatch
                    F8 59  F5  063C        SOBGTR  R9,10$                       ; Do the whole buffer
                     0087  31  063F        BRW     25$                          ; Done the whole buffer
                             0642
                             0642    20$:  ; Output data compare error message
                             0642
                0183'CF  D6  0642          INCL    ERROR_COUNT                  ; Bump the error count
            55  01A9 C7  DE  0646          MOVAL   UETUNT$Q_DEVDSC(R7),R5       ; Get address of unit name descriptor
                             064B          $FAO_S  CTRSTR = DATA_ERR_MSG,-      ; prepare message
                             064B                  OUTLEN = BUFFER_PTR,-
                             064B                  OUTBUF = FAO_BUF,-
                             064B                  P1 = R5                      ; Unit name descriptor
                000C'CF  DF  0660          PUSHAL  BUFFER_PTR                   ; ...push error msg adr
            000F0001 8F  DD  0664          PUSHL   #^XF0001                     ; ...push arg count
            00741132 8F  DD  066A          PUSHL   #UETP$_TEXT!STS$K_ERROR      ; ...push signal name
                0183'CF  DD  0670          PUSHL   ERROR_COUNT                  ; ...and the error count...
                00E1'CF  DF  0674          PUSHAL  PROCESS_NAME                 ; ...our own name...
            00010002 8F  DD  0678          PUSHL   #^X10002                     ; ...and the argument count...
            00748022 8F  DD  067E          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
        00000000'GF   07  FB  0684         CALLS   #7,G^LIB$SIGNAL              ; ...and print the error
                             068B          $FAO_S  CTRSTR = DROP_UNIT_MSG,-     ; prepare message
                             068B                  OUTLEN = BUFFER_PTR,-
                             068B                  OUTBUF = FAO_BUF,-
                             068B                  P1 = R5                      ; Unit name descriptor
                000C'CF  DF  06A0          PUSHAL  BUFFER_PTR                   ; Dropped unit message
                     01  DD  06A4          PUSHL   #1                           ; Arg count
            00741132 8F  DD  06A6          PUSHL   #UETP$_TEXT!STS$K_ERROR      ; Msg code and severity
```

```
 00000000'GF      03    FB  06AC          CALLS   #3,G^LIB$SIGNAL           ; and print message
                  02    8A  06B3          BICB2   #UETUNT$M_TESTABLE,-      ; Mark unit untestable
             0B A7          06B5                  UETUNT$B_FLAGS(R7)
          01C6'CF       97  06B7          DECB    START_CNT                 ; No more testing for this unit!
                  52    14  06BB          BGTR    40$                       ; BR if there are still units running
                            06BD          $WAKE_S                           ; Wake up the start routine so testing
                            06C8                                            ; will end (no more units)
                        04  06C8          RET
                            06C9  25$:
     01C0 C7      04    80  06C9          ADDB    #4,UETUNT$B_BUFPTR(R7)    ; Set index for next buffer
     58   01C0 C7       9A  06CE          MOVZBL  UETUNT$B_BUFPTR(R7),R8    ; Get buffer list index
          03D5'C8       D5  06D3          TSTL    BUF_SZ_LIST(R8)           ; End of list?
                  19    12  06D7          BNEQ    30$                       ; If not branch
                            06D9          $CLOSE-                           ; End of this file
                            06D9                  FAB = UETUNT$K_FAB(R7),-
                            06D9                  SUC = AST_CLOSE,-
                            06D9                  ERR = RMS_ERROR
          01BA'CF       D6  06EC          INCL    ITERATION                 ; Count the files completed
                  1D    11  06F0          BRB     40$
                            06F2  30$:
   20 A6  03D5'C8       B0  06F2          MOVW    BUF_SZ_LIST(R8),RAB$W_USZ(R6) ; Set next size
   24 A6  01C2 C7       DE  06F8          MOVAL   UETUNT$K_RBUF(R7),RAB$L_UBF(R6) ; Set read buffer address
                            06FE          $READ-                            ; Read the next record
                            06FE                  RAB = (R6),-
                            06FE                  SUC = AST_READ,-
                            06FE                  ERR = RMS_ERROR
                        04  070F  40$:    RET
                            0710
                            0710  ; Entered from CLOSE function.  Starts a new file with CREATE, unless
                            0710  ; an end-of-pass timeout has occured, then it does a REWIND QIO.
                            0710
                            0710  AST_CLOSE:
                      0FFC  0710          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
      56    04 AC       D0  0712          MOVL    4(AP),R6                  ; Get FAB address
      57    18 A6       D0  0716          MOVL    FAB$L_CTX(R6),R7          ; Get unit block address
   3E 0002'CF     01    E1  071A          BBC     #TEST_OVERV,FLAG,10$      ; BR if normal time-out has not occured
                  01    E1  0720          BBC     #UETUNT$V_TESTABLE,-      ; If unit not testable quit trying
          4A 0B A7          0722                  UETUNT$B_FLAGS(R7),20$
      56    0160 C7     DE  0725          MOVAL   UETUNT$K_RAB(R7),R6       ; Get RAB address
                            072A          $ASSIGN_S-                        ; Get channel number for async rewind
                            072A                  DEVNAM = UETUNT$Q_DEVDSC(R7),-
                            072A                  CHAN = UETUNT$W_CHAN(R7)
                            073A          $QIO_S-                           ; Rewind to BOT
                            073A                  CHAN = UETUNT$W_CHAN(R7),-
                            073A                  FUNC = #IO$_REWIND!IO$M_NOWAIT,- ; Do it asycronously
                            073A                  ASTADR = AST_REWIND,-
                            073A                  ASTPRM = R6
                  11    11  075C          BRB     20$
                            075E  10$:
                            075E          $CREATE-                          ; Start a new file
                            075E                  FAB = (R6),-
                            075E                  SUC = AST_CREATE,-
                            075E                  ERR = RMS_ERROR
                        04  076F  20$:    RET
                            0770
                            0770  ; Entered from CREATE function.  Does a CONNECT to start writing again.
                            0770
```

```
                              0770  AST_CREATE:
                      0FFC    0770          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
    56   04 AC         D0     0772          MOVL    4(AP),R6                  ; Get FAB address
    57   18 A6         D0     0776          MOVL    FAB$L_CTX(R6),R7          ; Get unit block address
              01       E1     077A          BBC     #UETUNT$V_TESTABLE,-      ; If unit not testable quit trying
      1C 0B A7                077C                  UETUNT$B_FLAGS(R7),10$
    56   0160 C7       DE     077F          MOVAL   UETUNT$K_RAB(R7),R6       ; Get RAB address
01C0 C7    FC 8F       90     0784          MOVB    #-4,UETUNT$B_BUFPTR(R7) ; Initialize the buffer list pointer
                              078A          $CONNECT-                         ; Connect the RAB
                              078A                  RAB = (R6),-
                              078A                  SUC = AST_WRITE,-
                              078A                  ERR = RMS_ERROR
                        04    079B  10$:    RET
                              079C
                              079C  ; Entered from a REWIND QIO. Decremen^s the active count and issues a WAKE to
                              079C  ; the start routine.
                              079C
                              079C  AST_REWIND:
                      0FFC    079C          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
    56   04 AC         D0     079E          MOVL    4(AP),R6                  ; Get RAB address
    57   18 A6         D0     07A2          MOVL    RAB$L_CTX(R6),R7          ; Get unit block address
              01       E1     07A6          BBC     #UETUNT$V_TESTABLE,-      ; If unit not testable quit trying
      1A 0B A7                07A8                  UETUNT$B_FLAGS(R7),10$
                              07AB          $DASSGN_S-                        ; Release channel
                              07AB                  CHAN = UETUNT$W_CHAN(R7)
        01C6'CF        97     07B6          DECB    START_CNT                 ; Decrease active count
                              07BA          $WAKE_S                           ; Wake up the start routine
                        04    07C5  10$:    RET
                              07C6
                              07C6
                              07C6
```

```
                                        07C6
                                        07C6  ; ONE SHOT MODE - This routine performs synchronous QIO's to verify the
                                        07C6  ; testablity of each unit. A record of random data is written, read in reverse,
                                        07C6  ; compared and if  no errors are detected the unit is marked testable.
                                        07C6
                                        07C6  ONE_SHOT:
                                        07C6          $SETSFM_S ENBFLG = #0           ; Disable system service failure mode
     57  0200'CF  00000200'8F      C1   07CF          ADDL3   #UNIT_LIST,UNIT_LIST,R7 ; Set the unit block list header
                                        07D9
                                        07D9  ONESHOT_LOOP:   ; Repeat for each unit
                                        07D9
                           01      E0   07D9          BBS     #UETUNT$V_TESTABLE,-    ; If unit not testable skip to next one
                     03 0B A7           07DB                  UETUNT$B_FLAGS(R7),5$
                         0123      31   07DE          BRW     NEXT_UNIT
                                        07E1  5$:
                   0098'CF   57    D0   07E1          MOVL    R7,CUR_UNTBLK           ; Save address of current unit block
                                        07E6          $SETIMR_S DAYTIM = ONEMIN_DELTA,- ; Set a watch dog timer
                                        07E6                    EFN    = #2,-
                                        07E6                    ASTADR = UNIT_TIMEOUT ; Where we go if something hangs
                                        07F9          $ASSIGN_S-                      ; Assign a channel to the tape unit
                                        07F9                  DEVNAM = UETUNT$Q_DEVDSC(R7),-
                                        07F9                  CHAN   = UETUNT$W_CHAN(R7)
                   018B'CF   50    D0   0809          MOVL    R0,IOSTAT               ; Save return status code
                         012A      30   080E          BSBW    ERROR_CHECK             ; Check for errors
                                        0811
                                        0811  ; Create and access the file
                                        0811
                           10      88   0811          BISB2   #UETUNT$M_MODIFIED,-    ; Let's flag tape as modified before we
                        0B A7           0813                  UETUNT$B_FLAGS(R7)      ; ...do it in case we get an error
                                        0815          $QIOW_S CHAN = UETUNT$W_CHAN(R7),-
                                        0815                  FUNC = #IO$_CREATE!IO$M_ACCESS!IO$M_CREATE,-
                                        0815                  IOSB = IOSTAT,-         ; Address of I/O status word
                                        0815                  P1 = FIB_DESC,-         ; FIB descriptor
                                        0815                  P2 = #ONESHOT_DESC      ; Name descriptor
                         00FD      30   083B          BSBW    ERROR_CHECK             ; Check for errors
                                        083E
                                        083E  ; Write a block of random data
                                        083E
                                        083E          $QIOW_S CHAN = UETUNT$W_CHAN(R7),-
                                        083E                  FUNC = #IO$_WRITEVBLK,- ; Write virtual block
                                        083E                  IOSB = IOSTAT,-         ; Address of I/O status word
                                        083E                  P1   = @WRITE_BUF,-     ; Random data buffer
                                        083E                  P2   = #WRITE_SIZE      ; Byte count
                         00D6      30   0862          BSBW    ERROR_CHECK             ; Check for errors
                                        0865
                                        0865  ; Preform a space reverse zero blocks so that the ACP will allow read access.
                                        0865
                      037B'CF      D4   0865          CLRL    FIB+FIB$L_CNTRLVAL      ; Set up to space zero blocks
                           04      B0   0869          MOVW    #FIB$C_SPACE,-          ; Set up for space function
                      0379'CF           086B                  FIB+FIB$W_CNTRLFUNC
                                        086E          $QIOW_S CHAN = UETUNT$W_CHAN(R7),-
                                        086E                  FUNC = #IO$_ACPCONTROL,- ; Perform ACP control function
                                        086E                  IOSB = IOSTAT,-         ; Address of I/O status word
                                        086E                  P1   = FIB_DESC
                         00AA      30   088E          BSBW    ERROR_CHECK             ; Check for errors
                                        0891
                                        0891  ; Read the file in reverse
```

```
                             0891  1381
                             0891  1382           $QIOW_S CHAN = UETUNT$W_CHAN(R7),-
                             0891  1383                   FUNC = #IO$_READVBLK!IO$M_REVERSE,-
                             0891  1384                   IOSB = IOSTAT,-
                             0891  1385                   P1   = UETUNT$K_RBUF(R7),- ; Read buffer
                             0891  1386                   P2   = #WRITE_SIZE
             0081        30  08B7  1387           BSBW    ERROR_CHECK                 ; Check for errors
                             08BA  1388
                             08BA  1389 ; Compare data read to data written
                             08BA  1390
   59   00008000 8F      D0  08BA  1391           MOVL    #WRITE_SIZE,R9              ; Get size of buffers
      5A    01C2 C7      DE  08C1  1392           MOVAL   UETUNT$K_RBUF(R7),R10       ; Get read buffer
      5B    0210'DF      DE  08C6  1393           MOVAL   @WRITE_BUF,R11              ; Get write buffer
                             08CB  1394 10$:
            8B    8A     91  08CB  1395           CMPB    (R10)+,(R11)+               ; Check the byte
                  05     12  08CE  1396           BNEQ    20$                         ; BR if bytes are same
               F8 59     F5  08D0  1397           SOBGTR  R9,10$                      ; Go do next byte-until done
                  0E     11  08D3  1398           BRB     30$                         ; Data check complete
                             08D5  1399
                             08D5  1400 20$:      ; Data compare failed
                             08D5  1401
  0002'CF     0040 8F    A8  08D5  1402           BISW2   #DATA_ERRM,FLAG             ; Set data error flag
                018B'CF  7C  08DC  1403           CLRQ    IOSTAT                      ; Clear possible left over error code
                   0070  31  08E0  1404           BRW     REPORT_ERROR                ; Go report error
                             08E3  1405
                             08E3  1406 30$:      ; Data compare ok - deaccess the file
                             08E3  1407
                             08E3  1408           $QIOW_S CHAN = UETUNT$W_CHAN(R7),-      ; Deaccess the file
                             08E3  1409                   FUNC = #IO$_DEACCESS,-
                             08E3  1410                   IOSB = IOSTAT
                   0037  30  0901  1411           BSBW    ERROR_CHECK                 ; Check for errors
                             0904  1412
                             0904  1413 NEXT_UNIT: ; Do the next unit - if there is more
                             0904  1414
      50      0C A7      D0  0904  1415           MOVL    UETUNT$W_CHAN(R7),R0        ; Did we ever $ASSIGN this drive?
                     0D  13  0908  1416           BEQL    5$                          ; BR if not - need not $DASSGN
                             090A  1417           $DASSGN_S  CHAN = R0                ; Deassign the .hannel
                   0024  30  0914  1418           BSBW    ERROR_CHECK                 ; Check for er ors
                             0917  1419 5$:       $CANTIM_S                           ; Forget the watchdog timer
               57    67  C0  0920  1420           ADDL2   (R7),R7                     ; Next unit block
      00000200'8F    57  D1  0923  1421           CMPL    R7,#UNIT_LIST               ; Done all units?
                     0C  12  092A  1422           BNEQ    10$                         ; Go check next unit
                             092C  1423           $SETSFM_S ENBFLG = #1               ; Enable system service failure mode
                   00DC  31  0935  1424           BRW     END_PASS                    ; All done!
                             0938  1425 10$:
                   FE9E  31  0938  1426           BRW     ONESHOT_LOOP
                             093B  1427
                             093B  1428 ERROR_CHECK: ; Here we check for QIO errors
                             093B  1429
                  09 50  E8  093B  1430           BLBS    R0,10$                      ; BR if I/O request queued OK
                     50  D5  093E  1431           TSTL    R0                          ; Test R0 for return status of zero
                     11  13  0940  1432           BEQL    REPORT_ERROR                ; BR if zero
          018B'CF    50  D0  0942  1433           MOVL    R0,IOSTAT                   ; Set error code
                             0947  1434 10$:
                018B'CF  D5  0947  1435           TSTL    IOSTAT                      ; Have we a status here?
                     05  13  094B  1436           BEQL    20$                         ; BR if not
             01 018B'CF  E9  094D  1437           BLBC    IOSTAT,REPORT_ERROR         ; BR if QIO failed
```

```
                                 05  0952  1438 20$:    RSB                              ; Return to test - no errors detected
                                     0953  1439
                                     0953  1440 REPORT_ERROR: ; We got an error - output appropriate message(s)
                                     0953  1441
                            02   8A  0953  1442         BICB2   #UETUNT$M_TESTABLE,-     ; Mark unit untestable
                         0B A7       0955  1443                 UETUNT$B_FLAGS(R7)
                                     0957  1444         $CANTIM_S                        ; Forget the watchdog timer
                       0183'CF   D6  0960  1445         INCL    ERROR_COUNT              ; Bump the error count
                       018B'CF   DD  0964  1446         PUSHL   IOSTAT                   ; Push the error code...
                       018B'CF   DD  0968  1447         PUSHL   IOSTAT                   ; ...and the error code...
                       01A9 C7   DF  096C  1448         PUSHAL  UETUNT$Q_DEVDSC(R7)      ; ...and the device designation.
                       000F'CF   DF  0970  1449         PUSHAL  TEST_NAME                ; ...and the test name...
                   000F0003 8F   DD  0974  1450         PUSHL   #^XF0003                 ; ...and the arg count...
                   0074819A 8F   DD  097A  1451         PUSHL   #UETP$_DEUNUS!STS$K_ERROR ; ...and the signal name...
                       0183'CF   DD  0980  1452         PUSHL   ERROR_COUNT              ; ...and the error count...
                       00E1'CF   DF  0984  1453         PUSHAL  PROCESS_NAME             ; ...our own name...
                   00010002 8F   DD  0988  1454         PUSHL   #^X10002                 ; ...and the argument count...
                   00748022 8F   DD  098E  1455         PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
              00000000'GF   0A   FB  0994  1456         CALLS   #10,G^LIB$SIGNAL         ; ...and print the error
             0830 8F   018B'CF   B1  099B  1457         CMPW    IOSTAT,#SS$_CANCEL       ; Was IO canceled because of timeout?
                            07   13  09A2  1458         BEQL    10$                      ; BR if it was
                  2C   018B'CF   B1  09A4  1459         CMPW    IOSTAT,#SS$_ABORT        ; Was IO aborted because of timeout?
                            15   12  09A9  1460         BNEQ    20$                      ; BR if it wasn't
                                     09AB  1461
                                     09AB  1462 10$:    ; Something must have hung and watch dog timer went off
                                     09AB  1463
                       01E6'CF   DF  09AB  1464         PUSHAL  TIME_OUT_MSG             ; ...push the error message adr
                            01   DD  09AF  1465         PUSHL   #1                       ; ...push the arg count...
                   00741132 8F   DD  09B1  1466         PUSHL   #UETP$_TEXT!STS$K_ERROR  ; ...push the signal name...
              00000000'GF   03   FB  09B7  1467         CALLS   #3,G^LIB$SIGNAL          ; ...report the error...
                            3A   11  09BE  1468         BRB     30$
                                     09C0  1469
                                     09C0  1470 20$:    ; Data compare error? Output error msg if it was
                                     09C0  1471
               34 0002'CF   06   E1  09C0  1472         BBC     #DATA_ERRORV,FLAG,30$    ; BR if not data compare error
                  58   01A9 C7   DE  09C6  1473         MOVAL   UETUNT$Q_DEVDSC(R7),R8   ; Get address of unit name descriptor
                                     09CB  1474         $FAO_S  CTRSTR = DATA_ERR_MSG,-  ; prepare message
                                     09CB  1475                 OUTLEN = BUFFER_PTR,-
                                     09CB  1476                 OUTBUF = FAO_BUF,-
                                     09CB  1477                 P1 = R8                  ; Unit name descriptor
                       000C'CF   DF  09E0  1478         PUSHAL  BUFFER_PTR               ; ...push error msg adr
                            01   DD  09E4  1479         PUSHL   #1                       ; ...push arg count
                   00741132 8F   DD  09E6  1480         PUSHL   #UETP$_TEXT!STS$K_ERROR  ; ...push signal name
              00000000'GF   03   FB  09EC  1481         CALLS   #3,G^LIB$SIGNAL          ; ...report the error
             0002'CF   0040 8F   AA  09F3  1482         BICW2   #DATA_ERRM,FLAG          ; Clear flag - error has been printed
                                     09FA  1483 30$:
                       018B'CF   D4  09FA  1484         CLRL    IOSTAT                   ; Prevent possible confusion later
                          FF03   31  09FE  1485         BRW     NEXT_UNIT
                                     0A01  1486
                                     0A01  1487 UNIT_TIMEOUT: ; Go here with watchdog timer timeout
                                     0A01  1488
                               0000  0A01  1489         .WORD   0
                  56   0098'CF   D0  0A03  1490         MOVL    CUR_UNTBLK,R6               ; Get the unit block address
                                     0A08  1491         $CANCEL_S CHAN = UETUNT$W_CHAN(R6) ; This IO will never complete
                                 04  0A13  1492         RET
                                     0A14  1493
```

```
                                        0A14  1495
                                        0A14  1496 END_PASS:
                                        0A14  1497
                                        0A14  1498 ;  This routine is entered on completion of a pass. In normal and
                                        0A14  1499 ; one-shot modes the image exits. In loop mode  the end of pass message is
                                        0A14  1500 ; output, the units are dismount, initialized, remounted and another pass is
                                        0A14  1501 ; started.
                                        0A14  1502
                 53 0002'CF     05  E1  0A14  1503         BBC     #LOOP_MODEV,FLAG,10$     ; BR if not loop forever
                    0002'CF     02  AA  0A1A  1504         BICW2   #TEST_OVERM,FLAG         ; Reset the termination flag
                        01BE'CF     D6  0A1F  1505         INCL    PASS                     ; Bump the pass count
                                        0A23  1506         $FAO_S  CTRSTR = PASS_MSG,-      ; Format the end of pass msg
                                        0A23  1507                 OUTLEN = BUFFER_PTR,-
                                        0A23  1508                 OUTBUF = FAO_BUF,-
                                        0A23  1509                 P1     = PASS,-
                                        0A23  1510                 P2     = ITERATION,-
                                        0A23  1511                 P3     = #0
                        000C'CF     DF  0A40  1512         PUSHAL  BUFFER_PTR               ; Push the string desc.
                             01     DD  0A44  1513         PUSHL   #1                       ; Push arg count
                    00741133 8F     DD  0A46  1514         PUSHL   #UETP$_TEXT!STS$K_INFO   ; Push the signal name
              00000000'GF   03      FB  0A4C  1515         CALLS   #3,G^LIB$SIGNAL          ; Print the end of pass message
                        01BA'CF     D4  0A53  1516         CLRL    ITERATION                ; Reset the iteration count
                    0C2E'CF     00  FB  0A57  1517         CALLS   #0,DISMOUNT_TAPE         ; Let's go dismount the tape(s)
                             01     DD  0A5C  1518         PUSHL   #1                       ; Set loop mode
                    0D8C'CF     01  FB  0A5E  1519         CALLS   #1,INIT_TAPE             ; Let's go init the tape(s) we modified
                             01     DD  0A63  1520         PUSHL   #1                       ; Set loop mode
                    0A81'CF     01  FB  0A65  1521         CALLS   #1,MOUNT_TAPE            ; Go mount tape(s) for another pass
                           FA58     31  0A6A  1522         BRW     RESTART                  ; Do the next pass
                                        0A6D  1523 10$:
       0187'CF    10000001 8F       D0  0A6D  1524         MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                                        0A76  1525         $EXIT_S STATUS                   ; Exit with the status
                                        0A81  1526
                                        0A81                  .SBTTL  Mount Routine
                                        0A81  ;++
                                        0A81  ; FUNCTIONAL DESCRIPTION:
                                        0A81  ;               This routine calls the $MOUNT system service for each tape drive. If
                                        0A81  ;               the mount completes successfully the label and hardware write
                                        0A81  ;               protection are all checked. If the tape passes all the tests the
                                        0A81  ;               UETUNT$M_MOUNTED and UETUNT$M_TESTABLE flags are set and the density
                                        0A81  ;               is saved in UETUNT$K_DENSITY. If the tape fails any of the above tests
                                        0A81  ;               the unit is dismounted and an error message is output. In loop mode
                                        0A81  ;               if the UETUNT$M_TESTABLE flag is not set the unit is skipped.
                                        0A81  ;               If no units are mounted successfully an error message is printed and
                                        0A81  ;               ERROR_EXIT is called.
                                        0A81  ;
                                        0A81  ; CALLING SEQUENCE:
                                        0A81  ;               PUSHL   #0 or #1        ; 0 for startup, 1 for loop mode
                                        0A81  ;               CALLS   #1,MOUNT_TAPE
                                        0A81  ;
                                        0A81  ; INPUT PARAMETERS:
                                        0A81  ;               NONE
                                        0A81  ;
                                        0A81  ; IMPLICIT INPUTS:
                                        0A81  ;               UNIT_LIST points to the head of a doubly linked circular list of unit
                                        0A81  ;                         blocks for the device(s) under test.
                                        0A81  ;
                                        0A81  ; OUTPUT PARAMETERS:
                                        0A81  ;               NONE
                                        0A81  ;
                                        0A81  ; IMPLICIT OUTPUTS:
                                        0A81  ;               Error message if $MOUNT fails or tape label and hardware
                                        0A81  ;               protection is not correct.
                                        0A81  ;
                                        0A81  ; COMPLETION CODES:
                                        0A81  ;               NONE
                                        0A81  ;
                                        0A81  ; SIDE EFFECTS:
                                        0A81  ;               Image will exit if no units are mounted successfully.
                                        0A81  ;
                                        0A81  ;--
                                        0A81
                                        0A81  MOUNT_TAPE:
                                  0FFC  0A81                  .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                        0A83
                        55   04 AC  D0  0A83                  MOVL    4(AP),R5                ; Get mode argument
        57   0200'CF   00000200'8F  C1  0A87                  ADDL3   #UNIT_LIST,UNIT_LIST,R7 ; Set the unit block list header
                           01AF'CF  94  0A91                  CLRB    UNIT_CNT                ; Clear the number of testable units
                           01D7'CF  94  0A95                  CLRB    ARG_COUNT               ; Initialize the error arg count
                                        0A99
                                        0A99  MOUNT_LOOP: ; Return here for each unit
                                        0A99
                             08 55  E9  0A99                  BLBC    R5,10$                  ; BR if we are just starting up
                                01  E0  0A9C                  BBS     #UETUNT$V_TESTABLE,-    ; BR if unit is still testable
                          03 0B A7      0A9E                          UETUNT$B_FLAGS(R7),10$
                              0111  31  0AA1                  BRW     NEXT_UNT                ; This unit failed last pass - skip it
                                        0AA4  10$:
                           01A9 C7  D0  0AA4                  MOVL    UETUNT$Q_DEVDSC(R7),-   ; Setup device name length
                           009C'CF      0AA8                          DEVDSC
                           01A9 C7  B0  0AAB                  MOVW    UETUNT$Q_DEVDSC(R7),-   ; Also set device name length in mount
```

```
                     037F'CF        0AAF                     MNT_LIST                 ;    item list
           01B1 C7   009C'CF   28   0AB2          MOVC3      DEVDSC,UETUNT$K_DEV_NAM(R7),- ; Get the device name
                     00F8'CF        0AB9                     DEV_NAME
                                    0ABC          $MOUNT_S-                           ; This amounts to MOUNT/NOASSIST/OV=ID
                                    0ABC                     ITMLST = MNT_LIST
                     01     50 D1   0AC7          CMPL       R0,#SS$_NORMAL
                            20 13   0ACA          BEQL       20$                      ; BR if no errors
                            50 DD   0ACC          PUSHL      R0                       ; Set the error code
                                    0ACE          $FAO_S     CTRSTR = MNT_ERR_MSG,- ; prepare message
                                    0ACE                     OUTLEN = BUFFER_PTR,-
                                    0ACE                     OUTBUF = FAO_BUF,·
                                    0ACE                     P1     = #DEVDSC         ; Unit name descriptor
                            01 DD   0AE7          PUSHL      #1                       ; ...push partial arg count
                          00EC 31   0AE9          BRW        MNT_ERROR                ; Go tell everyone
                                    0AEC
                                    0AEC   20$:   ; Unit mounted ok - let's find out what we got
                                    0AEC
                                    0AEC          $GETDEV_S-                          ; Get info on this device
                                    0AEC                     DEVNAM = DEVDSC,-
                                    0AEC                     PRIBUF = DIB
                                    0B01
                                    0B01          , Here we verify the tape label
                                    0B01
                  56   012F'CF 9A   0B01          MOVZBL     DIBBUF+DIB$W_VOLNAMOFF,R6 ; Get volume name offset
  0110'C6    0055'CF   004D'CF 29   0B06          CMPC3      LABEL,LABEL+8,DIBBUF+1(R6) ; Check for correct label
                            2D 13   0B10          BEQL       30$                      ; BR if label is correct
             56    0000010F'8F C0   0B12          ADDL2      #DIBBUF,R6               ; Get address of label descriptor
                                    0B19          $FAO_S     CTRSTR = LABEL_ERR_MSG,- ; prepare message
                                    0B19                     OUTLEN = BUFFER_PTR,-
                                    0B19                     OUTBUF = FAO_BUF,-
                                    0B19                     P1     = #DEVDSC,-       ; Unit name descriptor
                                    0B19                     P2     = R6,-            ; Tape label
                                    0B19                     P3     = #LABEL          ; Expected label
                            00 DD   0B3A          PUSHL      #0                       ; ...push partial arg count
                          008C 31   0B3C          BRW        DISMNT                   ; We can't test this one - dismount it
                                    0B3F
                                    0B3F   30$:   ; Here we check to see if the unit is hardware write-locked
                                    0B3F
              1E 0117'CF    13 E1   0B3F          BBC        #MT$V_HWL,DIBBUF+DIB$L_DEVDEPEND,50$ ; BR if not write-locked
                                    0B45          $FAO_S     CTRSTR = HWL_ERR_MSG,- ; prepare message
                                    0B45                     OUTLEN = BUFFER_PTR,-
                                    0B45                     OUTBUF = FAO_BUF,-
                                    0B45                     P1     = #DEVDSC         ; Unit name descriptor
                            00 DD   0B5E          PUSHL      #0                       ; ...push partial arg count
                          0068 31   0B60          BRW        DISMNT                   ; We can't test this one - dismount it
                                    0B63
                                    0B63   50$:   ; Tape passed tests - let's get the density
                                    0B63
                     05     08 EF   0B63          EXTZV      #MT$V_DENSITY,#MT$S_DENSITY,- ; Get density field
                58   0117'CF        0B66                     DIBBUF+DIB$L_DEVDEPEND,R8
                05   03     58 8F   0B6A          CASEB      R8,#3,#5                 ; BR according to density
                            001C'   0B6E   60$:   .WORD      0800$-60$                ; 0800 bpi - MT$K_NRZI_800
                            0025'   0B70          .WORD      1600$-60$                ; 1600 bpi - MT$K_PE_1600
                            002E'   0B72          .WORD      6250$-50$                ; 6250 bpi - MT$K_GCR_6250
                                    0B74
                                    0B74   ; Case fell through, unrecognized density
                                    0B74
```

```
          000C'CF   02C2'CF   D0  0B74                MOVL    DENSITY_ERR,BUFFER_PTR  ; Move error msg to buffer
 0014'CF  02CA'CF   02C2'CF   28  0B7B                MOVC3   DENSITY_ERR,DENSITY_ERR+8,BUFFER
                         00   DD  0B85                PUSHL   #0
                       0041   31  0B87                BRW     DISMNT
                                  0B8A   0800$:
           59  000003ED'8F    D0  0B8A                MOVL    #NRZI,R9                ; Get address of density 800
                         10   11  0B91                BRB     70$
                                  0B93   1600$:
           59  000003F2'8F    D0  0B93                MOVL    #PE,R9                  ; Get address of density 1600
                         07   11  0B9A                BRB     70$
                                  0B9C   6250$:
           59  000003F7'8F    D0  0B9C                MOVL    #GCR,R9                 ; Get address of density 6250
                                  0BA3
                                  0BA3   70$:     ; If we made it here we have what looks like a testable unit
                                  0BA3
                     69  05   28  0BA3                MOVC3   #DENS_LEN,(R9),-        ; Save density
                    01A4 C7       0BA6                        UETUNT$K_DENSITY(R7)
                         02   88  0BA9                BISB2   #UETUNT$M_TESTABLE,-    ; Mark unit testable
                      0B A7       0BAB                        UETUNT$B_FLAGS(R7)
                         08   88  0BAD                BISB2   #UETUNT$M_MOUNTED,-     ; Set mounted flag
                      0B A7       0BAF                        UETUNT$B_FLAGS(R7)
                    01AF'CF   96  0BB1                INCB    UNIT_CNT                ; Bump the unit count
                                  0BB5
                                  0BB5   NEXT_UNT: ; Do next unit - if there is more
                                  0BB5
                     57  67   C0  0BB5                ADDL2   (R7),R7                 ; Get next unit block
           00000200'8F   57   D1  0BB8                CMPL    R7,#UNIT_LIST           ; End of list?
                         03   13  0BBF                BEQL    10$                     ; BR if end
                       FED5   31  0BC1                BRW     MOUNT_LOOP
                                  0BC4   10$:
                    01AF'CF   D5  0BC4                TSTL    UNIT_CNT                ; Any units to test?
                         4F   13  0BC8                BEQL    MOUNT_EXIT              ; BR if no units mounted
                              04  0BCA                RET
                                  0BCB
                                  0BCB   DISMNT: ; here we dismount the untestable units
                                  0BCB
                                  0BCB                $DISMOU_S-                      ; Dismount and let it unload
                                  0BCB                        DEVNAM = UETUNT$Q_DEVDSC(R7)
                                  0BD8
                                  0BD8   MNT_ERROR: ; If we got here we have a unit in trouble - tell everyone and go on
                                  0BD8              ; to next unit
                                  0BD8
          01D7'CF    07  8E   C1  0BD8                ADDL3   (SP)+,#7,ARG_COUNT      ; Get total # args, pop partial count
                    0183'CF   D6  0BDE                INCL    ERROR_COUNT             ; Keep running error count
                    000C'CF   DF  0BE2                PUSHAL  BUFFER_PTR              ; Get error msg
                000F0001 8F   DD  0BE6                PUSHL   #^XF0001                ; ...argument count...
                00741132 8F   DD  0BEC                PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name...
                    0183'CF   DD  0BF2                PUSHL   ERROR_COUNT             ; Finish off arg list...
                    00E1'CF   DF  0BF6                PUSHAL  PROCESS_NAME            ; ...our own name...
                00010002 8F   DD  0BFA                PUSHL   #^X10002                ; ...
                00748022 8F   DD  0C00                PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
      00000000'GF   01D7'CF   FB  0C06                CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                         02   8A  0C0F                BICB2   #UETUNT$M_TESTABLE,-    ; Mark unit untestable - we could be in
                      0B A7       0C11                        UETUNT$B_FLAGS(R7)      ; ... loop mode
                    01D7'CF   94  0C13                CLRB    ARG_COUNT               ; Initialize the error arg count
                         9C   11  0C17                BRB     NEXT_UNT
                                  0C19
```

```
                 0C19  1699 MOUNT_EXIT:  ; If no testable units it's time to bail out
                 0C19  1700
     0165'CF  DF 0C19  1701         PUSHAL  NOUNIT_TESTABLE          ; Get error msg
          01  DD 0C1D  1702         PUSHL   #1                       ; ...argument count...
 00741132 8F  DD 0C1F  1703         PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name...
          03  DD 0C25  1704         PUSHL   #3                       ; ...and parameter count
     0187'CF  D4 0C27  1705         CLRL    STATUS                   ; We already said enough
        064A  31 0C2B  1706         BRW     ERROR_EXIT               ; ...and give up, complaining
                 0C2E  1707
                 0C2E  1708
```

```
                           0C2E        .SBTTL  Dismount Routine
                           0C2E  ;++
                           0C2E  ; FUNCTIONAL DESCRIPTION:
                           0C2E  ;       This routine checks the UETUNT$M_MOUNTED flag for each tape drive and
                           0C2E  ;       if it is set the $DISMOUNT system service is called with the nounload
                           0C2E  ;       qualifier. When the dismount is complete the UETUNT$M_MOUNTED flag is
                           0C2E  ;       cleared.
                           0C2E  ;
                           0C2E  ; CALLING SEQUENCE:
                           0C2E  ;       CALLS   #0,DISMOUNT_TAPE
                           0C2E  ;
                           0C2E  ; INPUT PARAMETERS:
                           0C2E  ;       NONE
                           0C2E  ;
                           0C2E  ; IMPLICIT INPUTS:
                           0C2E  ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                           0C2E  ;                 blocks for the device under test.
                           0C2E  ;
                           0C2E  ; OUTPUT PARAMETERS:
                           0C2E  ;       NONE
                           0C2E  ;
                           0C2E  ; IMPLICIT OUTPUTS:
                           0C2E  ;       NONE
                           0C2E  ;
                           0C2E  ; COMPLETION CODES:
                           0C2E  ;       NONE
                           0C2E  ;
                           0C2E  ; SIDE EFFECTS:
                           0C2E  ;       NONE
                           0C2E  ;
                           0C2E  ;--
                           0C2E
                           0C2E  DISMOUNT_TAPE:
                     0FFC  0C2E        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                           0C30
57   0200'CF   00000200'8F   C1  0C30        ADDL3   #UNIT_LIST,UNIT_LIST,R7 ; Set the unit block list header
                           0C3A        $SETAST_S ENBFLG = #1            ; Enable AST delivery
          0197'CF    D4  0C43        CLRL    AST_MODE                 ; Assume it was disabled
          09    50   D1  0C47        CMPL    R0,#SS$_WASSET           ; Were AST's enabled?
                05   12  0C4A        BNEQ    10$                      ; BR if not enabled
     0197'CF    01   D0  0C4C        MOVL    #1,AST_MODE              ; Set it to be reenabled
                         0C51  10$:  $SETSFM_S ENBFLG = #0            ; Disable SS failure mode
          019B'CF    D4  0C5A        CLRL    SS_FAIL_MODE             ; Assume it was disabled
          09    50   D1  0C5E        CMPL    R0,#SS$_WASSET           ; Was SS failure mode enabled?
                05   12  0C61        BNEQ    DISMNT_LOOP              ; BR if not enabled
     019B'CF    01   D0  0C63        MOVL    #1,SS_FAIL_MODE          ; Set it to be reenabled
                         0C68
                         0C68  DISMNT_LOOP: ; Return here for each unit
                         0C68
                03   E0  0C68        BBS     #UETUNT$V_MOUNTED,-      ; BR if tape is mounted
          03 0B A7       0C6A                UETUNT$B_FLAGS(R7),5$
             00AA    31  0C6D        BRW     NEXT1                    ; Skip to next unit
                         0C70  5$:
                         0C70        $DISMOU_S-                       ; Dismount/nounload
                         0C70                DEVNAM = UETUNT$W_DEVDSC(R7),-
                         0C70                FLAGS  = #DMT$M_NOUNLOAD
          01    50   D1  0C7D        CMPL    R0,#SS$_NORMAL           ; Dismount ok?
```

```
                 58    13  0C80  1767           BEQL    10$                         ; BR if no errors
        018B'CF  50    D0  0C82  1768           MOVL    R0,IOSTAT                   ; Set error code
                 02    8A  0C87  1769           BICB2   #UETUNT$M_TESTABLE,-        ; Mark unit untestable
              0B A7        0C89  1770                   UETUNT$B_FLAGS(R7)
           0183'CF     D6  0C8B  1771           INCL    ERROR_COUNT                 ; Bump the error count
           018B'CF     DD  0C8F  1772           PUSHL   IOSTAT                      ; Push the error code...
        58    01A9 C7  DE  0C93  1773           MOVAL   UETUNT$Q_DEVDSC(R7),R8  ; Get address of unit name descriptor
                           0C98  1774           $FAO_S  CTRSTR = DISMNT_ERR_MSG,- ; prepare message
                           0C98  1775                   OUTLEN = BUFFER_PTR,-
                           0C98  1776                   OUTBUF = FAO_BUF,-
                           0C98  1777                   P1     = R8             ; Unit name descriptor
           000C'CF     DF  0CAD  1778           PUSHAL  BUFFER_PTR              ; ...push error msg adr
      000F0001 8F      DD  0CB1  1779           PUSHL   #^XF0001                ; ...push arg count
      00741132 8F      DD  0CB7  1780           PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...push signal name
           0183'CF     DD  0CBD  1781           PUSHL   ERROR_COUNT             ; ...and the error count...
           00E1'CF     DF  0CC1  1782           PUSHAL  PROCESS_NAME            ; ...our own name...
      00010002 8F      DD  0CC5  1783           PUSHL   #^X10002                ; ...and the argument count...
      00748022 8F      DD  0CCB  1784           PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
     00000000'GF  08   FB  0CD1  1785           CALLS   #8,G^LIB$SIGNAL         ; ...and print the error
                  40   11  0CD8  1786           BRB     NEXT1
                           0CDA  1787
                           0CDA  1788 10$:      ; Here we set a watch dog timer and wait for dismount to complete
                           0CDA  1789
                           0CDA  1790           $SETIMR_S DAYTIM = THREEMIN_DELTA,- ; Set a three minute watch dog timer.
                           0CDA  1791                     EFN    = #2,-
                           0CDA  1792                     ASTADR = DISMOUNT_TIMEOUT     ; Where we go if something hangs
        0098'CF   57   D0  0CED  1793           MOVL    R7,CUR_UNTBLK           ; Save the unit block address
                           0CF2  1794 20$:
                           0CF2  1795           $GETDEV_S-                      ; Get info on this device
                           0CF2  1796                   DEVNAM = UETUNT$Q_DEVDSC(R7),-
                           0CF2  1797                   PRIBUF = DIB
    E5 010F'CF    13   E0  0D07  1798           BBS     #DEV$V_MNT,DIBBUF+DIB$L_DEVCHAR,20$ ; BR if still mounted
                           0D0D  1799           $CANTIM_S                       ; Cancel watch dog timer
                  08   8A  0D16  1800           BICB2   #UETUNT$M_MOUNTED,-     ; Clear mounted flag
               0B A7       0D18  1801                   UETUNT$B_FLAGS(R7)
                           0D1A  1802
                           0D1A  1803 NEXT1:    ; Do next unit - if there is more
                           0D1A  1804
                  57   67  C0  0D1A  1805       ADDL2   (R7),R7                 ; Get next unit block
     00000200'8F  57   D1  0D1D  1806           CMPL    R7,#UNIT_LIST           ; End of list?
                  03   13  0D24  1807           BEQL    10$                     ; BR if end
                FF3F   31  0D26  1808           BRW     DISMNT_LOOP             ; Go do next unit
                           0D29  1809 10$:      $SETSFM_S ENBFLG = SS_FAIL_MODE ; Set to previous state
                           0D34  1810           $SETAST_S ENBFLG = AST_MODE     ; Set to previous state
                       04  0D3F  1811           RET                             ; All done
                           0D40  1812
                           0D40  1813 DISMOUNT_TIMEOUT: ; We get here if dismount doesn't finish within three minutes
                           0D40  1814
                     0000  0D40  1815           .WORD   0
        56   0098'CF   D0  0D42  1816           MOVL    CUR_UNTBLK,R6           ; Get the unit block address
  018B'CF   0000022C 8F  D0  0D47  1817         MOVL    #SS$_TIMEOUT,IOSTAT     ; Set exit code
           018B'CF     DD  0D50  1818           PUSHL   IOSTAT
        58    01A9 C6  DE  0D54  1819           MOVAL   UETUNT$Q_DEVDSC(R6),R8  ; Get address of unit name descriptor
                           0D59  1820           $FAO_S  CTRSTR = DISMNT_ERR_MSG,- ; prepare message
                           0D59  1821                   OUTLEN = BUFFER_PTR,-
                           0D59  1822                   OUTBUF = FAO_BUF,-
                           0D59  1823                   P1     = R8             ; Unit name descriptor
```

```
                        0D6E  1824          $CANEXH_S                          ; Cancel all exit handlers
          000C'CF   DF  0D77  1825          PUSHAL  BUFFER_PTR                 ; ...push error msg adr
      000F0001 8F   DD  0D7B  1826          PUSHL   #^XF0001                   ; ...push arg count
      00741132 8F   DD  0D81  1827          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...push signal name
                03  DD  0D87  1828          PUSHL   #3
              04EC  31  0D89  1829          BRW     ERROR_EXIT
```

```
                                 0D8C         .SBTTL  Initialize Routine
                                 0D8C ;++
                                 0D8C ; FUNCTIONAL DESCRIPTION:
                                 0D8C ;       This routine initializes each tape drive in which the UETUNT$M_MODIFIED
                                 0D8C ;       flag is set. A DCL command file is created containing an INITIALIZE
                                 0D8C ;       command and is then executed as a subprocess. This is repeated for each
                                 0D8C ;       unit. In loop mode the tapes are initialized on subseqent passes
                                 0D8C ;       to different densities selected by rotating though a list of supported
                                 0D8C ;       densities. When testing is complete the tapes are initialized to their
                                 0D8C ;       orginally density.
                                 0D8C ;
                                 0D8C ; CALLING SEQUENCE:
                                 0D8C ;       PUSHL   #0 or #1        ; 1 for loop mode - 0 for all others
                                 0D8C ;       CALLS   #1,INIT_TAPE
                                 0D8C ;
                                 0D8C ; INPUT PARAMETERS:
                                 0D8C ;       Argument on stack for loop mode or other.
                                 0D8C ;
                                 0D8C ; IMPLICIT INPUTS:
                                 0D8C ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                                 0D8C ;                 blocks for the device under test.
                                 0D8C ;
                                 0D8C ; OUTPUT PARAMETERS:
                                 0D8C ;       NONE
                                 0D8C ;
                                 0D8C ; IMPLICIT OUTPUTS:
                                 0D8C ;       Command file MAGTAPE.COM is created temporarily. A subprocess is
                                 0D8C ;       temporarily created.
                                 0D8C ;
                                 0D8C ; COMPLETION CODES:
                                 0D8C ;       NONE
                                 0D8C ;
                                 0D8C ; SIDE EFFECTS:
                                 0D8C ;       If this routine aborts before it completes MAGTAPE.COM may be left
                                 0D8C ;       on the disk.
                                 0D8C ;
                                 0D8C ;--
                                 0D8C
                                 0D8C INIT_TAPE:
                                 0D8C
                            0FFC 0D8C         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0D8E
              55   04 AC    D0   0D8E         MOVL    4(AP),R5                   ; Get mode argument
57   0200'CF   00000200'8F  C1   0D92         ADDL3   #UNIT_LIST,UNIT_LIST,R7 ; Set the unit block list header
                                 0D9C         $SETAST_S ENBFLG = #1              ; Enable AST delivery
             0197'CF        D4   0DA5         CLRL    AST_MODE                   ; Assume it was disabled
             09     50      D1   0DA9         CMPL    R0,#SS$_WASSET             ; Were AST's enabled?
                    05      12   0DAC         BNEQ    10$                        ; BR if not enabled
       0197'CF      01      D0   0DAE         MOVL    #1,AST_MODE                ; Set it to be reenabled
                                 0DB3 10$:
                                 0DB3         $SETSFM_S ENBFLG = #0              ; Disable SS failure mode
             019B'CF        D4   0DBC         CLRL    SS_FAIL_MODE               ; Assume it was disabled
             09     50      D1   0DC0         CMPL    R0,#SS$_WASSET             ; Was SS failure mode enabled?
                    05      12   0DC3         BNEQ    INIT_LOOP                  ; BR if not enabled
       019B'CF      01      D0   0DC5         MOVL    #1,SS_FAIL_MODE            ; Set it to be reenabled
                                 0DCA
                                 0DCA INIT_LOOP: ; Return here for each unit
```

```
                          0DCA
                04   E0   0DCA           BBS     #UETUNT$V_MODIFIED,-       ; Init tape only if we wrote on it
          03 0B A7        0DCC                   UETUNT$B_FLAGS(R7),10$
              0164   31   0DCF           BRW     NEXT                       ; Skip to next unit
                          0DD2  10$:
                          0DD2           $CREATE-                           ; Create the command file
                          0DD2                   FAB = CMD_FAB
              01C3   30   0DDD           BSBW    ERR_CHK                    ; Go check for errors
                          0DE0           $CONNECT-                          ; Connect the selected RAB
                          0DE0                   RAB = INIT_RAB
              01B5   30   0DEB           BSBW    ERR_CHK                    ; Go check for errors
           56   0B   D0   0DEE           MOVL    #INIT_LEN,R6               ; Get init command length
                55   D5   0DF1           TSTL    R5                         ; Loop mode?
                0A   12   0DF3           BNEQ    20$                        ; BR if loop mode
      01A4 C7   05   28   0DF5           MOVC3   #DENS_LEN,UETUNT$K_DENSITY(R7),- ; Get original density
           0232'C6        0DFA                   CMD_BUF(R6)
                1E   11   0DFD           BRB     40$
                          0DFF
                          0DFF  20$:     ; Get new density for next pass if loop mode
                          0DFF
      01C1 C7   05   80   0DFF           ADDB2   #DENS_LEN,UETUNT$B_DENSPTR(R7)  ; Move index pointer to next density
      5A   01C1 C7   9A   0E04           MOVZBL  UETUNT$B_DENSPTR(R7),R10 ; Get the index
           03ED'CA   D5   0E09           TSTL    DENS_LIST(R10)             ; Terminator?
                06   12   0E0D           BNEQ    30$                        ; BR if it isn't
           01C1 C7   94   0E0F           CLRB    UETUNT$B_DENSPTR(R7)       ; Point to start of list
                5A   D4   0E13           CLRL    R10
                          0E15  30$:
      03ED'CA   05   28   0E15           MOVC3   #DENS_LEN,DENS_LIST(R10),- ; Fill in density in command string
           0232'C6        0E1A                   CMD_BUF(R6)
                          0E1D
                          0E1D  40$:     ; finish command string
                          0E1D
           56   05   C0   0E1D           ADDL2   #DENS_LEN,R6               ; Update length
           01A9 C7   28   0E20           MOVC3   UETUNT$Q_DEVDSC(R7),-      ; Fill in device name
   0232'C6 01B1 C7        0E24                   UETUNT$K_DEV_NAM(R7),CMD_BUF(R6)
        56 01A9 C7   A0   0E2A           ADDW2   UETUNT$Q_DEVDSC(R7),R6     ; Udate length
      022C'CF   06   28   0E2F           MOVC3   #LABEL_LEN,LABEL_CMD,-     ; Fill in the label
           0232'C6        0E34                   CMD_BUF(R6)
   0626'CF 56   06   C1   0E37           ADDL3   #LABEL_LEN,R6,-            ; Update length and put it in RAB
                          0E3D                   INIT_RAB+RAB$W_RSZ
                          0E3D
                          0E3D  ; Write command string to file
                          0E3D
                          0E3D           $PUT    RAB = INIT_RAB
              0158   30   0E48           BSBW    ERR_CHK                    ; Go check for errors
                          0E4B
                          0E4B  ; Close command file
                          0E4B
                          0E4B           $CLOSE  FAB = CMD_FAB
              014A   30   0E56           BSBW    ERR_CHK                    ; Go check for errors
                          0E59
                          0E59  ; Create a termination mailbox
                          0E59
    3D 0002'CF   08   E0  0E59           BBS     #MBX_CREATEDV,FLAG,50$     ; BR if mbx already exists
                          0E5F           $CREMBX_S-
                          0E5F                   CHAN = MBX_CHAN
              012E   30   0E72           BSBW    ERR_CHK                    ; Go check for errors
```

```
                              0E75            $GETCHN_S-                        ; Get its unit number
                              0E75                    CHAN = MBX_CHAN,-
                              0E75                    PRIBUF = DIB
               0115    30     0E8B            BSBW    ERR_CHK                   ; Go check for errors
    0359'CF   011B'CF  B0     0E8E            MOVW    DIBBUF+DIB$W_UNIT,MBX_UNIT      ; Save mbx unit number
    0002'CF   0100 8F  A8     0E95            BISW2   #MBX_CREATEDM,FLAG        ; set mbx created flag
                              0E9C
                              0E9C    ; now get the base priority of the parent process
                              0E9C
                              0E9C    50$:    $GETJPI_S ITMLST = GET_LIS,-
                              0E9C                      EFN    = #1
                              0EB1            $WAITFR_S EFN = #1                ; wait till this is done
                              0EBA
                              0EBA    ; Run command file as a subprocess
                              0EBA
                              0EBA            $CREPRC_S-
                              0EBA                    IMAGE  = LOGINOUT,-
                              0EBA                    INPUT  = CMD_FILE,-
                              0EBA                    OUTPUT = CMD_OUT,-
                              0EBA                    BASPRI = BASPRI,-
                              0EBA                    MBXUNT = MBX_UNIT
               00BA    30     0EE6            BSBW    ERR_CHK                   ; Go check for errors
                              0EE9
                              0EE9    ; It shouldn't take more than 30 seconds to complete
                              0EE9
                              0EE9            $SETIMR_S DAYTIM = THIRTYSEC_DELTA,- ; Set a thirty second watch dog timer.
                              0EE9                    EFN      = #2,-
                              0EE9                    ASTADR    = INIT_TIMEOUT
               00A4    30     0EFC            BSBW    ERR_CHK                   ; Go check for errors
                              0EFF            $QIOW_S-                          ; Wait for process to finish
                              0EFF                    CHAN = MBX_CHAN,-
                              0EFF                    EFN  = #1,-
                              0EFF                    FUNC = #IO$_READVBLK,-
                              0EFF                    IOSB = IOSTAT,-
                              0EFF                    P1   = MBX_BUF,-
                              0EFF                    P2   = #MBX_SIZE
               0044    30     0F24            BSBW    QIO_ERROR                 ; Subprocess complete ok?;
                 10    CA     0F27            BICL2   #UETUNT$M_MODIFIED,-      ; Clear modified flag
               0B A7          0F29                    UETUNT$B_FLAGS(R7)
                              0F2B
                              0F2B    ERASE:  ; Delete the temporary command file
                              0F2B
                              0F2B            $ERASE  FAB = CMD_FAB
                              0F36
                              0F36    NEXT:   ; Do next unit - if there is more
                              0F36
            57   67    C0     0F36            ADDL2   (R7),R7                   ; Get next unit block
    00000200'8F  57    D1     0F39            CMPL    R7,#UNIT_LIST             ; End of list?
                 03    13     0F40            BEQL    10$                       ; BR if end
               FE85    31     0F42            BRW     INIT_LOOP
                              0F45    10$:
            0183'CF    D5     0F45            TSTL    ERROR_COUNT               ; Any errors?
                 09    13     0F49            BEQL    20$                       ; BR if none
 0187'CF   10000002 8F D0     0F4B            MOVL    #STS$K_ERROR!STS$M_INHIB_MSG,STATUS ; Set exit code
                              0F54    20$:
                              0F54            $SETSFM_S ENBFLG = SS_FAIL_MODE ; Set to previous state
                              CF5F            $SETAST_S ENBFLG = AST_MODE     ; Set to previous state
```

```
                               04   0F6A          RET                               ; All done
                                    0F6B
                                    0F6B   QIO_ERROR: ; Here we check for QIO errors
                                    0F6B
                        04 50  E8   0F6B          BLBS    R0,10$                    ; BR if I/O request queued ok
                           50  D5   0F6E          TSTL    R0                        ; Test R0 for a return status of zero
                           07  13   0F70          BEQL    20$
                                    0F72   10$:
              01    018B'CF  B1   0F72          CMPW    IOSTAT,#SS$_NORMAL        ; I/O  successful?
                           07  13   0F77          BEQL    30$                       ; BR if error
                                    0F79   20$:
              50    018B'CF  D0   0F79          MOVL    IOSTAT,R0                 ; Set error code
                           23  11   0F7E          BRB     ERR_CHK                   ; Go print error
                                    0F80   30$:
                                    0F80          $CANTIM_S                         ; If we got here we don't need timer
                                    0F89                                            ; ....any longer
                     025B'CF  D1   0F89          CMPL    MBX_BUF+ACC$L_FINALSTS,- ; Check subprocess exit status
                          01        0F8D                  #SS$_NORMAL
                          01   12   0F8E          BNEQ    40$                       ; BR if error
                          05        0F90          RSB                               ; Return to test - no errors detected
                                    0F91   40$:
                     025B'CF  D0   0F91          MOVL    MBX_BUF+ACC$L_FINALSTS,- ; Set error code
                     018B'CF        0F95                  IOSTAT
       018B'CF   10000000 8F  CA   0F98          BICL2   #STS$M_INHIB_MSG,IOSTAT ; Clear inhibit msg bit
                          16   11   0FA1          BRB     OUTPUT_ERR
                                    0FA3
                                    0FA3   ERR_CHK: ; We come here to check for system service and RMS errors
                                    0FA3
                   01     50  D1   0FA3          CMPL    R0,#SS$_NORMAL            ; System service normal return?
                          10   13   0FA6          BEQL    10$
           00000000'8F    50  D1   0FA8          CMPL    R0,#RMS$_NORMAL           ; RMS normal return?
                          07   13   0FAF          BEQL    10$
              018B'CF    50  D0   0FB1          MOVL    R0,IOSTAT                 ; Set error code
                          01   11   0FB6          BRB     OUTPUT_ERR
                               05   0FB8   10$:   RSB
                                    0FB9
                                    0FB9   OUTPUT_ERR:     ; Sigh, something went wrong - better tell everyone
                                    0FB9
                          02   8A   0FB9          BICB2   #UETUNT$M_TESTABLE,-      ; Mark unit untestable
                       0B A7        0FBB                  UETUNT$B_FLAGS(R7)
                     0183'CF   D6   0FBD          INCL    ERROR_COUNT               ; Bump the error count
                     018B'CF   DD   0FC1          PUSHL   IOSTAT                    ; Push the error code...
              58    01A9 C7   DE   0FC5          MOVAL   UETUNT$Q_DEVDSC(R7),R8   ; Get address of unit name descriptor
                                    0FCA          $FAO_S  CTRSTR = INIT_ERR_MSG,- ; prepare message
                                    0FCA                  OUTLEN = BUFFER_PTR,-
                                    0FCA                  OUTBUF = FAO_BUF,-
                                    0FCA                  P1     = R8               ; Unit name descriptor
                     000C'CF   DF   0FDF          PUSHAL  BUFFER_PTR                ; ...push error msg adr
                  000F0001 8F   DD   0FE3          PUSHL   #^XF0001                  ; ...push arg count
                  00741132 8F   DD   0FE9          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...push signal name
                     0183'CF   DD   0FEF          PUSHL   ERROR_COUNT               ; ...and the error count...
                     00E1'CF   DF   0FF3          PUSHAL  PROCESS_NAME              ; ...our own name...
                  00010002 8F   DD   0FF7          PUSHL   #^X10002                  ; ...and the argument count...
                  00748022 8F   DD   0FFD          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
           00000000'GF    08   FB   1003          CALLS   #8,G^LIB$SIGNAL           ; ...and print the error
                        FF1E   31   100A          BRW     ERASE
                                    100D
           100D  2059 INIT_TIMEOUT:   , We get here if subprocess doesn't complete in 30 seconds
           100D  2060
     0FFC  100D  2061          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
           100F  2062
           100F  2063          $CANCEL_S CHAN = MBX_CHAN       ; This IO will never complete
       04  101B  2064          RET
           101C  2065
```

```
                         101C        .SBTTL  Timer Expiration Routine
                         101C  ;++
                         101C  ; FUNCTIONAL DESCRIPTION:
                         101C  ;       This routine will be called from a $SETIMR timeout.
                         101C  ;       This could be the normal timer to end the pass, or the
                         101C  ;       timer set to check for hung devices.
                         101C  ;
                         101C  ; CALLING SEQUENCE:
                         101C  ;       Called via AST at $SETIMR expiration.
                         101C  ;
                         101C  ; INPUT PARAMETERS:
                         101C  ;       REGIDT value in AST parameter.
                         101C  ;
                         101C  ; IMPLICIT INPUTS:
                         101C  ;       NONE
                         101C  ;
                         101C  ; OUTPUT PARAMETERS:
                         101C  ;       Done flag set for pass termination.
                         101C  ;
                         101C  ; IMPLICIT OUTPUTS:
                         101C  ;       NONE
                         101C  ;
                         101C  ; COMPLETION CODES:
                         101C  ;       Timeout status if device hung error.
                         101C  ;
                         101C  ; SIDE EFFECTS:
                         101C  ;       Sets a flag to indicate timer expiration.
                         101C  ;
                         101C  ;--
                         101C
                         101C  TIME_OUT:
                  0FFC   101C          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                         101E
        04 AC    01  D1  101E          CMPL    #REQIDT1,4(AP)          ; Is this the pass timer?
                 32  13  1022          BEQL    20$                     ; BR if yes
                         1024
                         1024  ; Some thing must have hung - let's try to cancel it
                         1024
                 00  DD  1024          PUSHL   #0                      ; Run down of image and indirect I/O
            01DB'CF  DF  1026          PUSHAL  RMSRUNDWN_BUF           ; Buffer to receive device & file
                         102A                                          ;  name of improperly closed files
     00000000'GF  02 FB  102A          CALLS   #2,G^SYS$RMSRUNDWN
            01C6'CF  97  1031          DECB    START_CNT               ; No more testing for this unit
                 0C  14  1035          BGTR    10$                     ; BR if there are still units running
                         1037          $WAKE_S                         ; Wake main routine
                     04  1042          RET
                         1043
                         1043  10$:    ; Set timer again in case something else hangs
                         1043
                         1043          $SETIMR_S-
                         1043                  DAYTIM = THIRTYSEC_DELTA,-
                         1043                  EFN    = #2,-
                         1043                  ASTADR = TIME_OUT,-
                         1043                  REQIDT = #REQIDT2       ; Hung device ID
                     04  1055          RET
                         1056
                         1056  20$:    ; Set test over flag and start a watch dog timer
```

```
                      1056
0002'CF    02   A8    1056         BISW2   #TEST_OVERM,FLAG         ; Ready to stop gracefully
                      105B         $SETIMR_S-                       ; Set timer after first rewind is
                      105B                 DAYTIM = THIRTYSEC_DELTA,- ; completed to check for device hung
                      105B                 EFN    = #2,-
                      105B                 ASTADR = TIME_OUT,-
                      105B                 REQIDT = #REQIDT2        ; Hung device ID
                      106D         $WAKE_S                          ; Wake startup routine
                04    1078         RET
                      1079
```

```
        .SBTTL  System Service Exception Handler
;++
; FUNCTIONAL DESCRIPTION:
;       This routine is executed if a software or hardware exception occurs or
;       if a LIB$SIGNAL system service is used to output a message.
;
; CALLING SEQUENCE:
;       Entered via an exception from the system
;
; INPUT PARAMETERS:
;       ERROR_COUNT   = previous cumulative error count
;                       -----------------
;          AP ---->     !       2       !
;                       !---------------!
;                       ! SIGNL ARY PNT !
;                       !---------------!
;                       ! MECH  ARY PNT !
;                       !---------------!   ---------
;                       !       4       !           ^
;                       !---------------!           !
;                       ! ESTABLISH FP  !           !
;                       !---------------!           !
;                       !     DEPTH     ! Mechanism Array
;                       !---------------!           !
;                       !      R0       !           !
;                       !---------------!           !
;                       !      R1       !           v
;                       !---------------!   ---------
;                       !       N       !           ^
;                       !---------------!           !
;                       ! CONDITION NAME!           !
;                       !---------------!           !
;                       ! N-3 ADDITIONAL!    Signal Array
;                       ! LONG WORD ARGS!           !
;                       !---------------!           !
;                       !      PC       !           !
;                       !---------------!           !
;                       !      PSL      !           v
;                       !---------------!   ---------
; IMPLICIT INPUTS:
;       NONE
;
; OUTPUT PARAMETERS:
;       NONE
;
; IMPLICIT OUTPUTS:
;       NONE
;
; COMPLETION CODES:
;       SS$_NORMAL if it's a UETP condition or RMS error.
;       Error status from exception, otherwise.
;
; SIDE EFFECTS:
;       May branch to ERROR_EXIT.
;       May print a message.
;--

```

```
                                1079  2192 SSERROR:
                        0FFC    1079  2193         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                107B  2194
                                107B  2195         $SETAST_S ENBFLG = #0           ; Disable AST delivery
                   01     DD    1084  2196         PUSHL   #1                      ; Assume ASTs were enabled
            50     09     D1    1086  2197         CMPL    S^#SS$_WASSET,R0        ; Were ASTs enabled?
                   02     13    1089  2198         BEQL    10$                     ; BR if they were
                   6E     D4    108B  2199         CLRL    (SP)                    ; Set ASTs to remain disabled
                                108D  2200 10$:
                                108D  2201         $SETSFM_S ENBFLG = #0           ; Disable SS failure mode
                   01     DD    1096  2202         PUSHL   #1                      ; Assume SS failure mode was enabled
            50     09     D1    1098  2203         CMPL    S^#SS$_WASSET,R0        ; Was SS failure mode enabled?
                   02     13    109B  2204         BEQL    20$                     ; BR if it was
                   6E     D4    109D  2205         CLRL    (SP)                    ; Set SS failure mode to remain off
                                109F  2206 20$:
       56     04 AC       D0    109F  2207         MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
       59     04 A6       7D    10A3  2208         MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
                   10     ED    10A7  2209         CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
                   0C           10A9  2210                 #STS$S_FAC_NO,-
  00000074 8F      59           10AA  2211                 R9,#UETP$_FACILITY
                   14     12    10B0  2212         BNEQ    30$                     ; BR if this is not a UETP exception
            66     02     C2    10B2  2213         SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                                10B5  2214         $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                   21     11    10C4  2215         BRB     40$                     ; Restore ASTs and SS fail mode
                                10C6  2216 30$:
  59    0000045C 8F       D1    10C6  2217         CMPL    #SS$_SSFAIL,R9          ; RMS failures are SysSvc failures
                   32     12    10CD  2218         BNEQ    50$                     ; BR if this can't be an RMS failure
                   10     ED    10CF  2219         CMPZV   #STS$V_FAC_NO,-         ; Is it an RMS failure?
                   0C           10D1  2220                 #STS$S_FAC_NO,-
            01     5A           10D2  2221                 R10,#RMS$_FACILITY
                   2B     12    10D4  2222         BNEQ    50$                     ; BR if not
  5A    F0000000 8F       CA    10D6  2223         BICL2   #^XF0000000,R10         ; Strip control bits from status code
       08 A6       04     39    10DD  2224         MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                   14           10E1  2225                 #NRAT_LENGTH,-
            0061'CF             10E2  2226                 NO_RMS_AST_TABLE        ; ...no AST can be delivered?
                   1A     13    10E5  2227         BEQL    50$                     ; BR if so - must give error here
                                10E7  2228 40$:
                   01     BA    10E7  2229         POPR    #^M<R0>                 ; Restore SS failure mode...
                                10E9  2230         $SETSFM_S ENBFLG = R0           ; ...
                   01     BA    10F2  2231         POPR    #^M<R0>                 ; Restore AST enable...
                                10F4  2232         $SETAST_S ENBFLG = R0           ; ...
            50     01     D0    10FD  2233         MOVL    S^#SS$_NORMAL,R0        ; Supply a standard status for exit
                   04           1100  2234         RET                             ; Resume processing (or goto RMS_ERROR)
                                1101  2235 50$:
       0187'CF     59     D0    1101  2236         MOVL    R9,STATUS               ; Save the status
                   58     D4    1106  2237         CLRL    R8                      ; Assume for now it's not SS failure
  59    0000045C 8F       D1    1108  2238         CMPL    #SS$_SSFAIL,R9          ; But is it a System Service failure?
                   38     12    110F  2239         BNEQ    70$                     ; BR if not - no special case message
                                1111  2240         $GETMSG_S MSGID = R10,-         ; Get SS failure code associated text
                                1111  2241                   MSGLEN = BUFFER_PTR,-
                                1111  2242                   BUFADR = FAO_BUF,-
                                1111  2243                   FLAGS  = #14,-
                                1111  2244                   OUTADR = MSG_BLOCK
            01C3'CF       95    1128  2245         TSTB    MSG_BLOCK+1             ; Get FAO arg count for SS failure code
                   16     13    112C  2246         BEQL    60$                     ; Don't use $GETMSG if no $FAO args...
            000C'CF       DF    112E  2247         PUSHAL  BUFFER_PTR              ; ...else build up...
                   01     DD    1132  2248         PUSHL   #1                      ; ...a message describing...
```

```
          00741130 8F   DD  1134  2249          PUSHL   #UETP$_TEXT                 ; ...why the System Service failed
                00 5A   F0  113A  2250          INSV    R10,#STS$V_SEVERITY,-       ; Give the message...
                6E 03       113D  2251                  #STS$S_SEVERITY,(SP)        ; ...the correct severity code
                58 03   D0  113F  2252          MOVL    #3,R8                       ; Count the number of args we pushed
                   05   11  1142  2253          BRB     70$
                            1144  2254 60$:
                   5A   DD  1144  2255          PUSHL   R10                         ; Save SS failure code
                58 01   D0  1146  2256          MOVL    #1,R8                       ; Count the number of args we pushed
                            1149  2257 70$:
           57   66 04   C5  1149  2258          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                5E 57   C2  114D  2259          SUBL2   R7,SP                         ; Save the current signal array...
       6E    04 A6 57   28  1150  2260          MOVC3   R7,CHF$L_SIG_NAME(P6),(SP) ; ...on the stack
           7E   66 58   C1  1155  2261          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                 011C   31  1159  2262          BRW     ERROR_EXIT
                            115C  2263
```

```
                          115C          .SBTTL  RMS Error Handler
                          115C    ;+
                          115C    ; FUNCTIONAL DESCRIPTION:
                          115C    ;       This routine handles error returns from RMS calls. If an error occurs
                          115C    ;       before testing is started the error codes are pushed on the stack
                          115C    ;       and  control is transfered to ERROR_EXIT. If an error occurs during
                          115C    ;       unit startup an error message is output and control is returned
                          115C    ;       to the startup routine. If testing is in progress when an error occurs
                          115C    ;       an error message is printed for the failing unit and the unit is marked
                          115C    ;       untestable. If additional units are still running we exit, otherwise
                          115C    ;       if there are units remaining to be started then control is returned to
                          115C    ;       the startup routine. If the failing unit is the last or only unit
                          115C    ;       running then a WAKE is queued and the pass timer is canceled.
                          115C    ;
                          115C    ; CALLING SEQUENCE:
                          115C    ;       Called by RMS when a file processing error is found.
                          115C    ;
                          115C    ; INPUT PARAMETERS:
                          115C    ;       The FAB or RAB associated with the RMS call.
                          115C    ;
                          115C    ; IMPLICIT INPUTS:
                          115C    ;       TEST_START - Test started flag
                          115C    ;       START_CNT - Count of units started
                          115C    ;       UNIT_CNT - Count of units to be started
                          115C    ;       UETUNT$M_TESTABLE - Unit testable flag in unit block
                          115C    ;
                          115C    ; OUTPUT PARAMETERS:
                          115C    ;       NONE
                          115C    ;
                          115C    ; IMPLICIT OUTPUTS:
                          115C    ;       Error message
                          115C    ;
                          115C    ; COMPLETION CODES:
                          115C    ;       NONE
                          115C    ;
                          115C    ; SIDE EFFECTS:
                          115C    ;       Program may exit, depending on were we are when the error occurs.
                          115C    ;
                          115C    ;--
                          115C
                          115C   RMS_ERROR:
                    0FFC  115C          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          115E
         56   04 AC   D0  115E          MOVL    4(AP),R6                    ; See whether we're dealing with...
            66   03   91  1162          CMPB    #FAB$C_BID,FAB$B_BID(R6)    ; ...a FAB or a RAB
                 11   12  1165          BNEQ    10$                         ; BR if it's a RAB
            58   56   D0  1167          MOVL    R6,R8                       ; ...add-ess of FAB...
              0C A6   DD  116A          PUSHL   FAB$L_STV(R6)               ; ...STV field for error...
              08 A6   DD  116D          PUSHL   FAB$L_STS(R6)               ; ...STS field for error...
    0187'CF   08 A6   D0  1170          MOVL    FAB$L_STS(R6),STATUS        ; ...and save the error code
                 10   11  1176          BRB     COMMON                      ; FAB and RAB share other code
                          1178   10$:
         58   3C A6   D0  1178          MOVL    RAB$L_FAB(R6),R8            ; ...address of associated FAB...
              0C A6   DD  117C          PUSHL   RAB$L_STV(R6)               ; ...STV field for error...
              08 A6   DD  117F          PUSHL   RAB$L_STS(R6)               ; ...STS field for error...
    0187'CF   08 A6   D0  1182          MOVL    RAB$L_STS(R6),STATUS        ; ...and save the error code
                          1188   COMMON:
```

```
      5A   34 A8   9A  1188  2322          MOVZBL  FAB$B_FNS(R8),R10      ; Get the file name size
                       118C  2323          $FAO_S  CTRSTR = RMS_ERR_MSG,- ; Common code, prepare error message.
                       118C  2324                  OUTLEN = BUFFER_PTR,-
                       118C  2325                  OUTBUF = FAO_BUF,-
                       118C  2326                  P1     = R10,-
                       118C  2327                  P2     = FAB$L_FNA(R8)
          000C'CF  DF  11A4  2328          PUSHAL  BUFFER_PTR                ; ...and arguments for ERROR_EXIT...
      000F0001 8F  DD  11A8  2329          PUSHL   #^XF0001                  ; ...
      00741130 8F  DD  11AE  2330          PUSHL   #UETP$_TEXT               ; ...
               00  EF  11B4  2331          EXTZV   #STS$V_SEVERITY,-
               03      11B6  2332                  #STS$S_SEVERITY,-
     59   0187'CF      11B7  2333                  STATUS,R9                 ; ...get the severity code...
          6E   59  88  11BB  2334          BISB2   R9,(SP)                   ; ...and add it into the signal name
  05 0002'CF   07  E0  11BE  2335          BBS     #TEST_STARTV,FLAG,10$     ; BR if testing in progress
               05  DD  11C4  2336          PUSHL   #5                        ; Current arg count
             00AF  31  11C6  2337          BRW     ERROR_EXIT                ; Time to bail-out
                       11C9  2338 10$:
          0183'CF  D6  11C9  2339          INCL    ERROR_COUNT               ; Update running error count
          0183'CF  DD  11CD  2340          PUSHL   ERROR_COUNT
          00E1'CF  DF  11D1  2341          PUSHAL  PROCESS_NAME
      00010002 8F  DD  11D5  2342          PUSHL   #^X10002
      00748020 8F  DD  11DB  2343          PUSHL   #UETP$_ERBOXPROC          ; Set the message code
          6E   59  88  11E1  2344          BISB2   R9,(SP)                   ; ...and the severity code
    00000000'GF 09 FB  11E4  2345          CALLS   #9,G^LIB$SIGNAL           ; Report error
      57   18 A6   D0  11EB  2346          MOVL    RAB$L_CTX(R6),R7          ; Get unit block address
     5A   01A9 C7  DE  11EF  2347          MOVAL   UETUNT$Q_DEVDSC(R7),R10 ; Get address of unit name descriptor
                       11F4  2348          $FAO_S  CTRSTR = DROP_UNIT_MSG,- ; prepare message
                       11F4  2349                  OUTLEN = BUFFER_PTR,-
                       11F4  2350                  OUTBUF = FAO_BUF,-
                       11F4  2351                  P1 = R10                  ; Unit name descriptor
          000C'CF  DF  1209  2352          PUSHAL  BUFFER_PTR                ; Dropped unit message
               01  DD  120D  2353          PUSHL   #1                        ; Arg count
      00741132 8F  DD  120F  2354          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Msg code and severity
    00000000'GF 03 FB  1215  2355          CALLS   #3,G^LIB$SIGNAL           ; Report message
               02  8A  121C  2356          BICB2   #UETUNT$M_TESTABLE,-      ; Mark unit untestable
            0B A7      121E  2357                  UETUNT$B_FLAGS(R7)
          01C6'CF  97  1220  2358          DECB    START_CNT                 ; No more testing for this unit
               1A  14  1224  2359          BGTR    20$                       ; BR if there are still units running
          01AF'CF  95  1226  2360          TSTB    UNIT_CNT                  ; Are there units not yet started?
               14  14  122A  2361          BGTR    20$                       ; BR if there are
                       122C  2362          $WAKE_S                           ; Wake up the start routine so testing
                       1237  2363                                            ; will end (no more units)
                       1237  2364          $CANTIM_S                         ; Cancel pass timer - we are all done
                   04  1240  2365 20$:     RET
                       1241  2366
                       1241  2367
```

```
                                 1241          .SBTTL  CTRL/C Handler
                                 1241  ;++
                                 1241  ; FUNCTIONAL DESCRIPTION:
                                 1241  ;       This routine handles CTRL/C AST's. It calls RMSRUNDWN to make sure
                                 1241  ;       there are no open files left around or any hung RMS I/O's pending.
                                 1241  ;
                                 1241  ; CALLING SEQUENCE:
                                 1241  ;       Called via AST
                                 1241  ;
                                 1241  ; INPUT PARAMETERS:
                                 1241  ;       NONE
                                 1241  ;
                                 1241  ; IMPLICIT INPUTS:
                                 1241  ;       NONE
                                 1241  ;
                                 1241  ; OUTPUT PARAMETERS:
                                 1241  ;       NONE
                                 1241  ;
                                 1241  ; IMPLICIT OUTPUTS:
                                 1241  ;       NONE
                                 1241  ;
                                 1241  ; COMPLETION CODES:
                                 1241  ;       NONE
                                 1241  ;
                                 1241  ; SIDE EFFECTS:
                                 1241  ;       NONE
                                 1241  ;
                                 1241  ;--
                                 1241
                                 1241  CCASTHAND:
                           0FFC  1241          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 1243
                                 1243  ; Output the abort message
                                 1243
             00B7'CF     DF      1243          PUSHAL  CNTRLCMSG                 ; Set message pointer
                  01     DD      1247          PUSHL   #1                        ; Set arg count
         00741130 8F     DD      1249          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                  00     DD      124F          PUSHL   #0                        ; Indicate an abnormal termination
             00E1'CF     DF      1251          PUSHAL  PROCESS_NAME              ; ...
                  02     DD      1255          PUSHL   #2                        ; ...
         007410E0 8F     DD      1257          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
   00000000'GF    07     FB      125D          CALLS   #7,G^LIB$SIGNAL           ; Output the message
                         D0      1264          MOVL    #<STS$M_INHIB_MSG!-       ; Set the exit status
                                 1265                  SS$_CONTROLC-=
                                 1265                  STS$K_SUCCESS+STS$K_WARNING>,-
0187'CF   10000650 8F            1265                  STATUS
                                 126D          $EXIT_S STATUS                    ; Terminate program cleanly
                                 1278
```

```
                                   1278          .SBTTL  Error Exit
                                   1278    ;++
                                   1278    ; FUNCTIONAL DESCRIPTION:
                                   1278    ;        This routine prints an error message and exits.
                                   1278    ;
                                   1278    ; CALLING SEQUENCE:
                                   1278    ;        MOVx  error status value,STATUS
                                   1278    ;        PUSHx error specific information on the stack
                                   1278    ;        PUSHL current argument count
                                   1278    ;        BRW   ERROR_EXIT
                                   1278    ;
                                   1278    ; INPUT PARAMETERS:
                                   1278    ;        Arguments to LIB$SIGNAL, as above
                                   1278    ;
                                   1278    ; IMPLICIT INPUTS:
                                   1278    ;        NONE
                                   1278    ;
                                   1278    ; OUTPUT PARAMETERS:
                                   1278    ;        Message to SYS$OUTPUT and SYS$ERROR
                                   1278    ;
                                   1278    ; IMPLICIT OUTPUTS:
                                   1278    ;        Program exit
                                   1278    ;
                                   1278    ; COMPLETION CODES:
                                   1278    ;        NONE
                                   1278    ;
                                   1278    ; SIDE EFFECTS:
                                   1278    ;        NONE
                                   1278    ;
                                   1278    ;--
                                   1278
                                   1278    ERROR_EXIT:
                                   1278
                                   1278          $SETAST_S ENBFLG = #0          ; ASTs can play havoc with messages
       15 0002'CF     03     E0    1281          BBS     #BEGIN_MSGV,FLAG,10$   ; BR if "begin" msg already printed
                      7E     D4    1287          CLRL    -(SP)                  ; Set the time stamp flag
             000F'CF         DF    1289          PUSHAL  TEST_NAME              ; Set the test name
                      02     DD    128D          PUSHL   #2                     ; Push the argument count
          00741039 8F        DD    128F          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
        00000000'GF   04     FB    1295          CALLS   #4,G^LIB$SIGNAL        ; Print the startup message
                                   129C    10$:
     01D7'CF    08    8E     C1    129C          ADDL3   (SP)+,#8,ARG_COUNT     ; Get total # args, pop partial count
             0183'CF         D6    12A2          INCL    ERROR_COUNT            ; Keep running error count
                      00     DD    12A6          PUSHL   #0                     ; Push the time parameter
             00E1'CF         DF    12A8          PUSHAL  PROCESS_NAME           ; Push test name...
          000F0002 8F        DD    12AC          PUSHL   #^XF0002               ; ...arg count...
          007410E2 8F        DD    12B2          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
             0183'CF         DD    12B8          PUSHL   ERROR_COUNT            ; Finish off arg list...
             00E1'CF         DF    12BC          PUSHAL  PROCESS_NAME           ; ...our own name...
          00010002 8F        DD    12C0          PUSHL   #^X10002               ; ...
          00748022 8F        DD    12C6          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
   00000000'GF   01D7'CF     FB    12CC          CALLS   ARG_COUNT,G^LIB$SIGNAL ; Truly bitch
                                   12D5
             0187'CF         D5    12D5          TSTL    STATUS                 ; Did we exit with an error code?
                      09     12    12D9          BNEQ    20$                    ; BR if we did
          007410E2 8F        D0    12DB          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
             0187'CF               12E1                  STATUS
```

```
                                    12E4  20$:
      0187'CF   1000000C 8F   C8    12E4         BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                                    12ED         $EXIT_S STATUS                  ; Exit in error
                                    12F8
```

```
                                  12F8          .SBTTL  Exit Handler
                                  12F8  ;++
                                  12F8  ; FUNCTIONAL DESCRIPTION:
                                  12F8  ;       This routine handles cleanup at exit.  If the MODE logical name is
                                  12F8  ;       equated to "ONE", this routine will update the test flag in the
                                  12F8  ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                                  12F8  ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                                  12F8  ;       under test. All mounted units will be dismounted and all modified
                                  12F8  ;       tapes will be initialized.
                                  12F8  ;
                                  12F8  ; CALLING SEQUENCE:
                                  12F8  ;       Invoked automatically by $EXIT System Service.
                                  12F8  ;
                                  12F8  ; INPUT PARAMETERS:
                                  12F8  ;       STATUS  contains the exit status.
                                  12F8  ;       FLAG    has synchronizing bits.
                                  12F8  ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                                  12F8  ;
                                  12F8  ; IMPLICIT INPUTS:
                                  12F8  ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                                  12F8  ;                 blocks for the device under test.
                                  12F8  ;
                                  12F8  ; OUTPUT PARAMETERS:
                                  12F8  ;       NONE
                                  12F8  ;
                                  12F8  ; IMPLICIT OUTPUTS:
                                  12F8  ;       Various files are de-accessed, the process name is reset, and any
                                  12F8  ;       necessary synchronization with UETPDEV01 is carried out.
                                  12F8  ;       If the MODE logical name is equated to "ONE", the routine will update
                                  12F8  ;       the test flag in the UETINIDEV.DAT file depending on the
                                  12F8  ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                                  12F8  ;       block for each unit for the device under test.
                                  12F8  ;
                                  12F8  ; COMPLETION CODES:
                                  12F8  ;       NONE
                                  12F8  ;
                                  12F8  ; SIDE EFFECTS:
                                  12F8  ;       NONE
                                  12F8  ;
                                  12F8  ;--
                                  12F8
                                  12F8  EXIT_HANDLER:
                            0FFC  12F8          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                  12FA
                                  12FA          $SETAST_S ENBFLG = #0           ; Disable AST delivery
                                  1303          $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
          0D 0002'CF    04    E0  130C          BBS     #ONESHOT_MODEV,FLAG,5$  ; Skip RMS run down if oneshot mode
                                  1312
                                  1312  ; Here we cancel any RMS I/O so the channels will be deassigned
                                  1312
                        00    DD  1312          PUSHL   #0                      ; Run down of image and indirect I/O
                   01DB'CF    DF  1314          PUSHAL  RMSRUNDWN_BUF           ; Buffer to receive device & file
                                  1318                                          ;  name of any improperly closed files
         00000000'GF    02    FB  1318          CALLS   #2,G^SYS$RMSRUNDWN
                                  131F  5$:
             F90A CF    00    FB  131F          CALLS   #0,DISMOUNT_TAPE        ; Let's go dismount the tape(s)
                        00    DD  1324          PUSHL   #0
          FA61 CF    01   FB 1326            CALLS   #1,INIT_TAPE                   ; Let's go init the tape(s) we modified
       06 0002'CF    04   E1 132B            BBC     #ONESHOT_MODEV,FLAG,10$ ; BR if not oneshot mode
       03 0002'CF    02   E0 1331            BBS     #SAFE_TO_UPDV,FLAG,20$  ; Is it ok to update?
              00CD   31      1337  10$:      BRW     END_UPDATE                     ; No updating to be done
                             133A
                             133A  20$:      ; See if we can find our place in uetinidev.dat
                             133A
          5A   0484'CF    DE 133A            MOVAL   INI_RAB,R10                    ; Set the RAB address
             1E AA   02   90 133F            MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
   10 AA   04C8'CF   06   28 1343            MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                             134A            $GET    RAB = (R10)                    ; Go back to the DDB record
                 03 50    E8 1353            BLBS    R0,30$                         ; BR  if successful
                  0090    31 1356            BRW     UPDATE_FAILED                  ; If failure then forget it
                             1359
                             1359  30$:      ; Let's  find out if we have any testable units
                             1359
             1E AA   00   90 1359            MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
 5B   0200'CF    00000200'8F  C1 135D        ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                     58   D4 1367            CLRL    R8                             ; Clear logical name flag
                     59   D4 1369            CLRL    R9                             ; Init a counter
                             136B
                             136B  UNIT_LOOP:        ; Return here until all units are checked
                             136B
                     01   E1 136B            BBC     #UETUNT$V_TESTABLE,-    ; BR if unit is not testable
               1B 0B AB      136D                    UETUNT$B_FLAGS(R11),10$
                     59   D6 1370            INCL    R9                             ; Count testable units
                     58   D5 1372            TSTL    R8                             ; Have we created logical name for RMS?
                     15   12 1374            BNEQ    10$                            ; BR if we have
                             1376
                             1376  ; If we have a testable unit tell the other tests about it
                             1376
                             1376            $CRELOG_S #1,LOGNAM_DESC,-      ; Create logical device name for
                             1376                     UETUNT$Q_DEVDSC(R11)   ; RMS test.
                     58   D6 1389            INCL    R8                             ; Set flag
                             138B
                             138B  10$:      ; Do next unit - if there is more
                             138B
                 5B  6B   C0 138B            ADDL2   (R11),R11                      ; Next unit block
       00000200'8F   5B   D1 138E            CMPL    R11,#UNIT_LIST                 ; Are we full circle in the list?
                     D4   12 1395            BNEQ    UNIT_LOOP                      ; BR if not
                     59   D5 1397            TSTL    R9                             ; Any testable units?
                     12   12 1399            BNEQ    20$                            ; BR if yes...
                             139B
                             139B  ; If no testable units mark the controller as untestable in uetinidev.dat
                             139B
       0018'CF    4E 8F   90 139B            MOVB    #^A/N/,BUFFER+4                ; ...else disable the DDB record...
                             13A1            $UPDATE RAB = (R10)                    ; ...here
                 3C 50    E9 13AA            BLBC    R0,UPDATE_FAILED               ; If error then forget it
                             13AD
                             13AD  20$:      ; We have testable unit(s) - update uetinidev.dat to reflect what we
                             13AD            ; found
                             13AD
                 5B  6B   C0 13AD            ADDL2   (R11),R11                      ; Next unit block
       00000200'8F   5B   D1 13B0            CMPL    R11,#UNIT_LIST                 ; Are we full circle in the list?
                     4E   13 13B7            BEQL    END_UPDATE                     ; BR if yes
                             13B9            $GET    RAB = (R10)                    ; Get a record
                 24 50    E9 13C2            BLBC    R0,UPDATE_FAILED               ; If error then forget it
```

```
      0014'CF   20    8A  13C5  2594              BICB2   #LC_BITM,BUFFER               ; Convert to uppercase
 0014'CF     55 8F    91  13CA  2595              CMPB    #^A/U/,BUFFER                 ; Is it a UCB record?
                35    12  13D0  2596              BNEQ    END_UPDATE                    ; BR if not
                01    E0  13D2  2597              BBS     #UETUNT$V_TESTABLE,-          ; BR if this unit is testable...
         D6 0B AB         13D4  2598                      UETUNT$B_FLAGS(R11),20$
 0018'CF     4E 8F    90  13D7  2599              MOVB    #^A/N/,BUFFER+4               ; ...else disable the UCB record...
                          13DD  2600              $UPDATE RAB = (R10)                   ; ...here
             C4 50    E8  13E6  2601              BLBS    R0,20$                        ; Look at the next record if no error
                          13E9  2602 UPDATE_FAILED:
             0C AA    DD  13E9  2603              PUSHL   RAB$L_STV(R10)                ; Do a simple message...
                50    DD  13EC  2604              PUSHL   R0                            ; ...to tell of the failure
          032A'CF     DF  13EE  2605              PUSHAL  INIDEV_UPDERR
                01    DD  13F2  2606              PUSHL   #1
                00    EF  13F4  2607              EXTZV   #STS$V_SEVERITY,-             ; Copy the severity from RMS status...
       7E    50 03        13F6  2608                      #STS$S_SEVERITY,R0,-(SP)
 6E   00741130 8F     C8  13F9  2609              BISL2   #UETP$_TEXT,(SP)              ; ...to our message
00000000'GF     05    FB  1400  2610              CALLS   #5,G^LIB$SIGNAL
                          1407  2611 END_UPDATE:
                          1407  2612
                          1407  2613 ; Output the ending message
                          1407  2614
                00    DD  1407  2615              PUSHL   #0                            ; Set the time flag
          000F'CF     DF  1409  2616              PUSHAL  TEST_NAME                     ; Push the test name
                02    DD  140D  2617              PUSHL   #2                            ; Push arg count
                00    EF  140F  2618              EXTZV   #STS$V_SEVERITY,-             ; Push the proper exit severity...
                03        1411  2619                      #STS$S_SEVERITY,-
       7E  0187'CF        1412  2620                      STATUS,-(SP)
 6E   00741080 8F     C8  1416  2621              BISL2   #UETP$_ENDEDD,(SP)            ; ...and use it in our message code
                04    DD  141D  2622              PUSHL   #4
          51    5E    D0  141F  2623              MOVL    SP,R1
                          1422  2624              $PUTMSG_S MSGVEC = (R1)               ; Output the message
                          1431  2625
                          1431  2626 ; Finish last minute clean up
                          1431  2627
                          1431  2628              $SETPRN_S PRCNAM = ACNT_NAME      ; Reset the process name
                      04  143C  2629              RET                               ; That's all folks!
                          143D  2630
                          143D  2631              .END    UETTAPE00
```

```
$$.TAB                          = 00000604 R      03    DIB                             00000107 R      03
$$.TABEND                       = 00000648 R      03    DIB$B_DEVCLASS                = 00000004
$$.TMP                          = 00000000              DIB$B_DEVTYPE                 = 00000005
$$.TMP1                         = 00000001              DIB$K_LENGTH                  = 00000074
$$.TMP2                         = 0000006A              DIB$L_DEVCHAR                 = 00000000
$$.TMPX                         = 00000023 R      04    DIB$L_DEVDEPEND               = 00000008
$$.TMPX1                        = 0000000B              DIB$W_UNIT                    = 0000000C
$$T1                            = 00000000              DIB$W_VOLNAMOFF               = 00000020
$$T2                            = 00000004              DIBBUF                          0000010F R      03
ACC$L_FINALSTS                  = 00000004              DISMNT                          00000BCB R      05
ACNT_NAME                         00000000 R      02    DISMNT_ERR_MSG                  00000306 R      02
ALL_SET                           000003FB R      05    DISMNT_LOOP                     00000C68 R      05
ARG_COUNT                         000001D7 R      03    DISMOUNT_TAPE                   00000C2E R      05
AST_CLOSE                         00000710 R      05    DISMOUNT_TIMEOUT                00000D40 R      05
AST_CREATE                        00000770 R      05    DMT$M_NOUNLOAD                = 00000001
AST_MODE                          00000197 R      03    DROP_UNIT_MSG                   00000394 R      02
AST_READ                          0000060F R      05    DUMMY_FAB                       00000520 R      03
AST_REWIND                        0000079C R      05    DUMMY_RAB                       00000570 R      03
AST_SPACE                         000005DE R      05    DVI$_DEVNAM                   = 00000020
AST_WRITE                         00000587 R      05    END_PASS                        00000A14 R      05
BASPRI                            00000193 R      03    END_UPDATE                      00001407 R      05
BEGIN_MSGM                      = 00000008              ERASE                           00000F2B R      05
BEGIN_MSGV                      = 00000003              ERROR_CHECK                     0000093B R      05
BUFFER                            00000014 R      03    ERROR_COUNT                     00000183 R      03
BUFFER_PTR                        0000000C R      03    ERROR_EXIT                      00001278 R      05
BUF_ADR_LIST                      00000218 R      03    ERR_CHK                         00000FA3 R      05
BUF_SZ_LIST                       000003D5 R      02    ESC                           = 0000001B
CCASTHAND                         00001241 R      05    EXIT_DESC                       000001C7 R      03
CHF$L_SIGARGLST                 = 00000004              EXIT_HANDLER                    000012F8 R      05
CHF$L_SIG_ARG1                  = 00000008              FAB$B_BID                     = 00000000
CHF$L_SIG_ARGS                  = 00000000              FAB$B_FNS                     = 00000034
CHF$L_SIG_NAME                  = 00000004              FAB$C_BID                     = 00000003
CMD_BUF                           00000232 R      03    FAB$C_BLN                     = 00000050
CMD_FAB                           000005B4 R      03    FAB$C_SEQ                     = 00000000
CMD_FILE                          0000043A R      02    FAB$C_VAR                     = 00000002
CMD_OUT                           0000042F R      02    FAB$L_ALQ                     = 00000010
CNTRLCMSG                         000000B7 R      02    FAB$L_CTX                     = 00000018
COMMON                            00001188 R      05    FAB$L_DEV                     = 00000040
CONTROLLER                        00000031 R      02    FAB$L_FNA                     = 0000002C
CONT_DESC                         0000036F R      02    FAB$L_FOP                     = 00000004
CS1                               00000096 R      02    FAB$L_STS                     = 00000008
CS3                               000000A8 R      02    FAB$L_STV                     = 0000000C
CUR_UNTBLK                        00000098 R      03    FAB$V_BRO                     = 00000006
DATA_ERRM                       = 00000040              FAB$V_CHAN_MODE               = 00000002
DATA_ERRORV                     = 00000006              FAB$V_CR                      = 00000001
DATA_ERR_MSG                      00000218 R      02    FAB$V_FILE_MODE               = 00000004
DDB_RFA                           000004C8 R      03    FAB$V_GET                     = 00000001
DEAD_CTPLNAME                     000000F8 R      02    FAB$V_LNM_MODE                = 00000000
DENSITY_ERR                       000002C2 R      02    FAB$V_PUT                     = 00000000
DENS_LEN                        = 00000005              FAB$V_UFO                     = 00000011
DENS_LIST                         000003ED R      02    FAB$V_UPD                     = 00000003
DEV$V_MNT                       = 00000013              FAB$V_UPI                     = 00000006
DEV$V_TRM                       = 00000002              FAB$W_GBC                     = 00000048
DEVDEP_SIZE                     = 0000001E              FAO_BUF                         00000004 R      03
DEVDSC                            0000009C R      03    FIB                             00000363 R      03
DEVNAM_LEN                        000001B0 R      03    FIB$C_SPACE                   = 00000004
DEV_NAME                          000000F8 R      03    FIB$L_CNTRLVAL                = 00000018
```

```
FIB$M_NOWRITE                  = 00000001                 MAX_UNIT_DESIG                  = 00000005
FIB$M_WRITE                    = 00000100                 MBX_BUF                           00000257 R     03
FIB$W_CNTRLFUNC                = 00000016                 MBX_CHAN                          00000357 R     03
FIB_DESC                         0000035B R     03        MBX_CREATEDM                    = 00000100
FIB_LEN                        = 0000001C                 MBX_CREATEDV                    = 00000008
FILE_SZ                        = 00000005                 MBX_SIZE                        = 00000100
FILNM                            000000D0 R     03        MBX_UNIT                          00000359 R     03
FILNM_DESC                       000000C8 R     03        MNT$M_NOASSIST                  = 00000004
FILNM_LEN                      = 00000009                 MNT$M_OVR_IDENT                 = 00000200
FIND_IT                          000001E1 R     05        MNT$_DEVNAM                     = 00000001
FLAG                             00000002 R     03        MNT$_FLAGS                      = 00000004
FOUND_IT                         00000279 R     05        MNT_ERROR                         00000BD8 R     05
GCR                              000003F7 R     02        MNT_ERR_MSG                       00000245 R     02
GET_LIS                          00000400 R     02        MNT_FLAGS                         0000039B R     03
HWL_ERR_MSG                      000002A5 R     02        MNT_LIST                          0000037F R     03
ILLEGAL_REC                      0000017F R     02        MODE                              00000041 R     02
INADDRESS                        0000019F R     03        MOUNT_EXIT                        00000C19 R     05
INIDEV_UPDERR                    0000032A R     02        MOUNT_LOOP                        00000A99 R     05
INIT_ERR_MSG                     000002E1 R     02        MOUNT_TAPE                        00000A81 R     05
INIT_LEN                       = 0000000B                 MSG_BLOCK                         000001C2 R     03
INIT_LOOP                        00000DCA R     05        MT$S_DENSITY                    = 00000005
INIT_RAB                         00000604 R     03        MT$V_DENSITY                    = 00000008
INIT_TAPE                        00000D8C R     05        MT$V_HWL                        = 00000013
INIT_TIMEOUT                     0000100D R     05        NAME_LEN                        = 0000000F
INI_FAB                          00000434 R     03        NEW_NODE                          00000208 R     03
INI_RAB                          00000484 R     03        NEXT                              00000F36 R     05
INPUT_ITMLST                     00000086 R     02        NEXT1                             00000D1A R     05
IO$M_ACCESS                    = 00000040                 NEXT_UNIT                         00000904 R     05
IO$M_CREATE                    = 00000080                 NEXT_UNT                          00000BB5 R     05
IO$M_CTRLCAST                  = 00000100                 NOUNIT_SELECTED                   0000013F R     02
IO$M_NOWAIT                    = 00000080                 NOUNIT_TESTABLE                   00000165 R     02
IO$M_REVERSE                   = 00000040                 NO_CTRLNAME                       000000D8 R     02
IO$_ACPCONTROL                 = 00000038                 NO_RMS_AST_TABLE                  00000061 R     02
IO$_CREATE                     = 00000033                 NRAT_LENGTH                     = 00000014
IO$_DEACCESS                   = 00000034                 NRZI                              000003ED R     02
IO$_READVBLK                   = 00000031                 ONEMIN_DELTA                      0000035B R     02
IO$_REWIND                     = 00000024                 ONESHOT_DESC                      000000B6 R     03
IO$_SETMODE                    = 00000023                 ONESHOT_LEN                     = 0000000A
IO$_WRITEVBLK                  = 00000030                 ONESHOT_LOOP                      000007D9 R     05
IOSTAT                           0000018B R     03        ONESHOT_MODEV                   = 00000004
ITERATION                        000001BA R     03        ONESHOT_MODM                    = 00000010
JPI$_PRIB                      = 00000309                 ONE_SHOT                          000007C6 R     05
LABEL                            0000004D R     02        OS_FILNM                          000000BE R     03
LABEL_CMD                        0000022C R     03        OUTADDRESS                        000001A7 R     03
LABEL_ERR_MSG                    00000266 R     02        OUTPUT_ERR                        00000FB9 R     05
LABEL_LEN                      = 00000006                 OUT_DEV                           00000437 R     02
LC_BITM                        = 00000020                 OUT_LEN                         = 00000003
LIB$SIGNAL                       ********   X   05        PAGES                           = 00000041
LOGINOUT                         00000410 R     02        PASS                              000001BE R     03
LOGNAM                           000000AC R     03        PASS_MSG                          000001B3 R     02
LOGNAM_DESC                      000000A4 R     03        PE                                000003F2 R     02
LOGNAM_LEN                     = 0000000A                 PMTSIZ                          = 0000C019
LOOP                             000004ED R     05        PROCESS_NAME                      000000E1 R     03
LOOP_MODEV                     = 00000005                 PROCESS_NAME_FREE               = 0000000B
LOOP_MODM                      = 00000020                 PROC_CONT_NAME                    0000008B R     05
MAX_DEV_DESIG                  = 0000000A                 PROMPT                            000003BC R     02
MAX_PROC_NAME                  = 0000000F                 QIO_ERROR                         00000F6B R     05
```

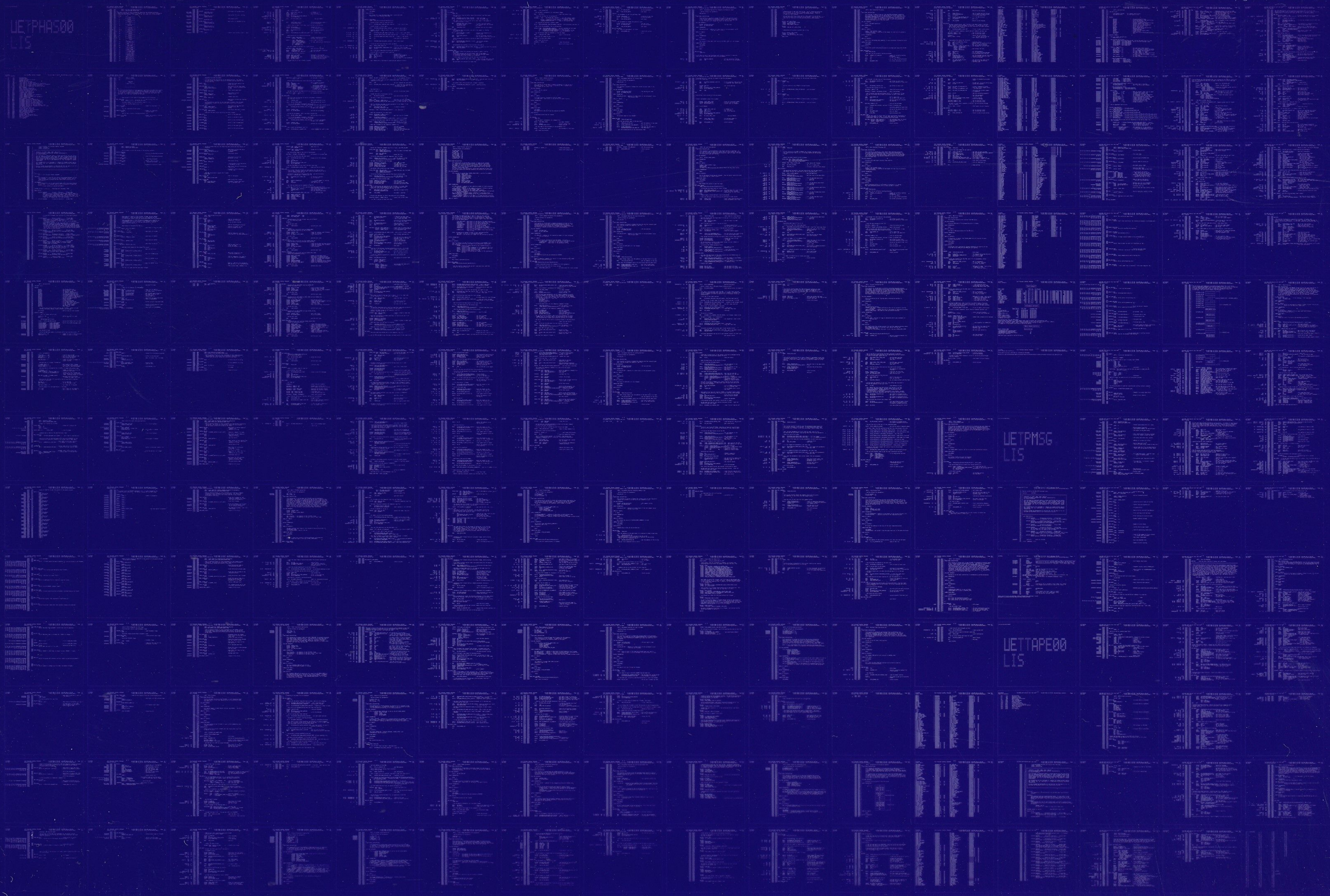
```
RAB$B_PSZ             = 00000034                 SS$_WASSET            = 00000009
RAB$B_RAC             = 0000001E                 SSERROR                 00001079 R     05
RAB$C_BID             = 00000001                 SS_FAIL_MODE            0000019B R     03
RAB$C_BLN             = 00000044                 SS_SYNCH_EFN          = 00000003
RAB$C_RFA             = 00000002                 START_CNT               000001C6 R     03
RAB$C_SEQ             = 00000000                 STATUS                  00000187 R     03
RAB$L_BKT             = 00000038                 STR$UPCASE              ********    X  05
RAB$L_CTX             = 00000018                 STS$K_ERROR           = 00000002
RAB$L_FAB             = 0000003C                 STS$K_INFO            = 00000003
RAB$L_PBF             = 00000030                 STS$K_SUCCESS         = 00000001
RAB$L_RBF             = 00000028                 STS$K_WARNING         = 00000000
RAB$L_ROP             = 00000004                 STS$M_INHIB_MSG       = 10000000
RAB$L_STS             = 00000008                 STS$S_FAC_NO          = 0000000C
RAB$L_STV             = 0000000C                 STS$S_SEVERITY        = 00000003
RAB$L_UBF             = 00000024                 STS$V_FAC_NO          = 00000010
RAB$V_ASY             = 00000000                 STS$V_SEVERITY        = 00000000
RAB$V_BIO             = 0000000B                 SUPDEV_GBLSEC           00000020 R     02
RAB$V_PMT             = 0000001E                 SUP_FAB                 000004D0 R     03
RAB$W_RFA             = 00000010                 SYS$ASSIGN              ********   GX  05
RAB$W_RSZ             = 00000022                 SYS$CANCEL              ********   GX  05
RAB$W_USZ             = 00000020                 SYS$CANEXH              ********   GX  05
RANDOM1                 000001B2 R     03        SYS$CANTIM              ********   GX  05
RANDOM2                 000001B6 R     03        SYS$CLOSE               ********   GX  05
READ_SIZE             = 00008000                 SYS$CONNECT             ********   GX  05
REC_SIZE              = 00000028                 SYS$CREATE              ********   GX  05
REPORT_ERROR            00000953 R     05        SYS$CRELOG              ********   GX  05
REQIDT1               = 00000001                 SYS$CREMBX              ********   GX  05
REQIDT2               = 00000002                 SYS$CREPRC              ********   GX  05
RESTART                 000004C5 R     05        SYS$CRMPSC              ********   GX  05
RMS$_BLN                ********    X  02        SYS$DASSGN              ********   GX  05
RMS$_BUSY               ********    X  02        SYS$DCLEXH              ********   GX  05
RMS$_CDA                ********    X  02        SYS$DISMOU              ********   GX  05
RMS$_FAB                ********    X  02        SYS$ERASE               ********   GX  05
RMS$_FACILITY         = 00000001                 SYS$EXIT                ********   GX  05
RMS$_NORMAL             ********    X  05        SYS$EXPREG              ********   GX  05
RMS$_RAB                ********    X  02        SYS$FAO                 ********    X  05
RMSRUNDWN_BUF           000001DB R     03        SYS$GET                 ********   GX  05
RMS_ERROR               0000115C R     05        SYS$GETCHN              ********   GX  05
RMS_ERR_MSG             00000377 R     02        SYS$GETDEV              ********   GX  05
RUNDWN_BUF              000001E3 R     03        SYS$GETDVI              ********   GX  05
SAFE_TO_UPDM          = 00000004                 SYS$GETJPI              ********   GX  05
SAFE_TO_UPDV          = 00000002                 SYS$GETMSG              ********   GX  05
SEC$M_EXPREG            ********    X  05        SYS$HIBER               ********   GX  05
SEC$M_GBL               ********    X  05        SYS$INPUT               00000075 R     02
SHR$_ABENDD           = 000010E0                 SYS$MGBLSC              ********   GX  05
SHR$_BEGIND           = 00001038                 SYS$MOUNT               ********   GX  05
SHR$_ENDEDD           = 00001080                 SYS$OPEN                ********   GX  05
SHR$_OPENIN           = 00001098                 SYS$PUT                 ********   GX  05
SHR$_TEXT             = 00001130                 SYS$PUTMSG              ********   GX  05
SS$_ABORT             = 0000002C                 SYS$QIO                 ********   GX  05
SS$_BADPARAM          = 00000014                 SYS$QIOW                ********   GX  05
SS$_CANCEL            = 00000830                 SYS$READ                ********   GX  05
SS$_CONTROLC          = 00000651                 SYS$RMSRUNDWN           ********    X  05
SS$_NORMAL            = 00000001                 SYS$SETAST              ********   GX  05
SS$_NOSUCHSEC         = 00000978                 SYS$SETIMR              ********   GX  05
SS$_SSFAIL            = 0000045C                 SYS$SETPRN              ********   GX  05
SS$_TIMEOUT           = 0000022C                 SYS$SETSFM              ********   GX  05
```

```
SYS$SPACE                          ********  GX    05        UNIT_LIST                        00000200 R      03
SYS$TRNLOG                         ********  GX    05        UNIT_LOOP                        0000136B R      05
SYS$UPDATE                         ********  GX    05        UNIT_TIMEOUT                     00000A01 R      05
SYS$WAITFR                         ********  GX    05        UPDATE_FAILED                    000013E9 R      05
SYS$WAKE                           ********  GX    05        WRITE_BUF                        00000210 R      03
SYS$WRITE                          ********  GX    05        WRITE_SIZE                     = 00008000
SYSIN_FAB                          000003A0 R      03
SYSIN_RAB                          000003F0 R      03
TEST_NAME                          0000000F R      02
TEST_OVERM                       = 00000002
TEST_OVERV                       = 00000001
TEST_STARTM                      = 00000080
TEST_STARTV                      = 00000007
TEXT_BUFFER                      = 00000084
THIRTYSEC                          0000034F R      02
THIRTYSEC_DELTA                    00000353 R      02
THREEMIN                           00000363 R      02
THREEMIN_DELTA                     00000367 R      02
TIME                               000000D9 R      03
TIME_OUT                           0000101C R      05
TIME_OUT_MSG                       000001E6 R      02
TTCHAN                             00000000 R      03
UETP                             = 00740000
UETP$_ABENDD                     = 007410E0
UETP$_ABORTC                     = 0074832B
UETP$_BEGIND                     = 00741038
UETP$_DENOSU                     = 00748335
UETP$_DEUNUS                     = 0074819A
UETP$_ENDEDD                     = 00741080
UETP$_ERBOXPROC                  = 00748020
UETP$_FACILITY                   = 00000074
UETP$_OPENIN                     = 00741098
UETP$_TEXT                       = 00741130
UETTAPE00                          00000000 RG     05
UETUNT$B_BUFPTR                  = 000001C0
UETUNT$B_DENSPTR                 = 000001C1
UETUNT$B_FLAGS                   = 0000000B
UETUNT$B_TYPE                    = 00000108
UETUNT$C_FAB                     = 00000110
UETUNT$C_INDSIZ                  = 000001A4
UETUNT$K_DENSITY                 = 000001A4
UETUNT$K_DEVDEP                  = 000001A4
UETUNT$K_DEV_NAM                 = 000001B1
UETUNT$K_FAB                     = 00000110
UETUNT$K_RAB                     = 00000160
UETUNT$K_RBUF                    = 000001C2
UETUNT$M_MODIFIED                = 00000010
UETUNT$M_MOUNTED                 = 00000008
UETUNT$M_TESTABLE                = 00000002
UETUNT$Q_DEVDSC                  = 000001A9
UETUNT$T_FILSPC                  = 00000014
UETUNT$V_MODIFIED                = 00000004
UETUNT$V_MOUNTED                 = 00000003
UETUNT$V_TESTABLE                = 00000001
UETUNT$W_CHAN                    = 0000000C
UETUNT$W_SIZE                    = 00000009
UNIT_CNT                           000001AF R      03
```

```
                                         +-----------------+
                                         ! Psect synopsis !
                                         +-----------------+

PSECT name                  Allocation          PSECT No.  Attributes
----------                  ----------          ---------  ----------
.  ABS  .                   00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                       00000000  (     0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
RODATA                      0000044D  (  1101.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC PAGE
RWDATA                      00000648  (  1608.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC PAGE
$RMSNAM                     0000002E  (    46.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
TAPE                        0000143D  (  5181.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE

                                     +------------------------+
                                     ! Performance indicators !
                                     +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                    29    00:00:00.09    00:00:01.06
Command processing               114    00:00:00.71    00:00:02.86
Pass 1                          1194    00:00:36.00    00:01:33.19
Symbol table sort                  0    00:00:03.67    00:00:09.07
Pass 2                          1066    00:00:09.67    00:00:33.07
Symbol table output                1    00:00:00.34    00:00:00.65
Psect synopsis output              1    00:00:00.03    00:00:00.03
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals            2408    00:00:50.53    00:02:19.97

The working set limit was 1500 pages.
194103 bytes (380 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2292 non-local and 94 local symbols.
2631 source lines were read in Pass 1, producing 46 object records in Pass 2.
84 pages of virtual memory were used to define 76 macros.

                                    +--------------------------+
                                    ! Macro library statistics !
                                    +--------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[UETP.OBJ]UETP.MLB;1                    2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      0
_$255$DUA28:[SYSLIB]STARLET.MLB;2                  71
TOTALS (all libraries)                             73

2670 GETS were required to define 73 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:UETTAPE00/OBJ=OBJ$:UETTAPE00 MSRC$:UETTAPE00/UPDATE=(ENH$:UETTAPE00)+EXECML$/LIB+LIB$:UETP/LIB
```

0412 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



UETPHAS00
LIS

UETPMSG
LIS

UETTAPE00
LIS

0413 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SATSSF04
MAP

SATSSF06
MAP

UETUNAS00
LIS

UETPSY

SATSSF01
MAP

SATSSF03
MAP

SATSSF05
MAP

SATSSF07
MAP

UETTTYS00
LIS

SATSSF02
MAP